On the Complexity of the Plantinga-Vegter Algorithm

Felipe Cucker, Alperen A. Ergür, Josué Tonelli-Cueto

On the Complexity of the Plantinga-Vegter Algorithm

Felipe Cucker · **Alperen A. Ergür** · **Josué Tonelli-Cueto**

Abstract We introduce a general toolbox for precision control and complexity analysis of subdivision based algorithms in computational geometry. We showcase the toolbox on a well known example from this family; the adaptive subdivision algorithm due to Plantinga and Vegter. The only existing complexity estimate on this rather fast algorithm was an exponential worst-case upper bound for its interval arithmetic version. We go beyond the worst-case by considering smoothed analysis, and prove polynomial time complexity estimates for both interval arithmetic and finite precision versions of the Plantinga-Vegter algorithm. The employed toolbox is a blend of robust probabilistic techniques coming from geometric functional analysis with condition numbers and the continuous amortization paradigm introduced by Burr, Krahmer and Yap. We hope this combination of tools from different disciplines would prove useful for understanding complexity aspects of the broad family of subdivision based algorithms in computational geometry.

Keywords Plantinga-Vegter algorithm, subdivision methods, complexity

Mathematics Subject Classification (2010) 65D18, 68W40

1 Introduction

Subdivision based algorithms are ubiquitous in computational geometry. These algorithms have the advantage of simplicity, and often have good practical performance. The two main challenges related to subdivision based algorithms are the control of precision (or a termination criterion), and complexity analysis. As late as summer 2019, complexity analysis aspect of subdivision based geometric algorithms was considered to be "largely open" [37]. In this paper, we contribute to both of the main challenges by introducing a hybrid toolbox that combines condition numbers, high dimensional probability theory, and continuous amortization framework introduced by Burr, Krahmer, and Yap [8]. To keep our writing focused, and the length of the article finite, we only showcase the toolbox on a well-known member of this large family; the algorithm of Plantinga and Vegter.

This work was supported by the Einstein Foundation Berlin. An extended abstract containing some of the results was presented at ISSAC'19 [14]. Some preliminary versions of the results in Section 8 was included in the doctoral thesis of J. Tonelli-Cueto [34].

F.C. was partially supported by a GRF grant from the Research Grants Council of the Hong Kong SAR (project number CityU 11302418).

J. T.-C. was partially supported by ANR JCJC GALOP (ANR-17-CE40-0009), the PGMO grant ALMA, and the PHC GRAPE.

F. Cucker
Dept. of Mathematics, City University of Hong Kong, HONG KONG. E-mail: macucker@cityu.edu.hk

A.A. Ergür
Carnegie Mellon University, School of Computer Science, Pittsburgh, PA, USA. E-mail: aergur@cs.cmu.edu

J. Tonelli-Cueto
Inria Paris & IMJ-PRG, Sorbonne Université, Paris, FRANCE. E-mail: josue.tonelli.cueto@bizkaia.eu

Plantinga-Vegter (PV) algorithm is an adaptive subdivison algorithm for meshing curves and surfaces [28]. The algorithm admits an implicit equation of a curve or a surface and outputs an isotopic piecewise linear approximation with controlled Hausdorff distance. The initial paper of Plantinga and Vegter contained no complexity analysis and not even a formal setting fixing either the kind of functions implicitly defining the considered curves and surfaces or the arithmetic used. However, concrete implementations in the paper indicated the efficiency of the algorithm. The algorithm is now widely considered to be very efficient.

The first complexity analysis of the PV algorithm was published thirteen years later by Burr, Gao and Tsigaridas [10] (cf. [11]). The paper of Burr, Gao, and Tsigaridas focused on the subdivision procedure of the Plantinga-Vegter algorithm and only analyzed the complexity for polynomials with integer coefficients. The paper provides bounds that are exponential both in the degree d of the input polynomial and in its logarithmic height τ. The discrepancy between the exponential complexity estimate and the practical efficiency of the PV algorithm was marked by the following comment at the end of the paper

> Even though our bounds are optimal, in practice, these are quite pessimistic [. . .]

The authors further observe that, following from their Proposition 5.2 (see Theorem 6.1 below) an instance-based analysis of the algorithm (i.e., one yielding a cost that depends on the input at hand) could be derived from the evaluation of a certain integral. And they conclude their paper by writing

> Since the complexity of the algorithm can be exponential in the inputs [size], the integral must be described in terms of additional geometric and intrinsic parameters.

In this paper, we make progress towards these aims by going beyond the worst-case analysis and by using condition numbers. We believe condition numbers are a perfect fit for the latter aim as they provide a geometric and an arguably intrinsic parameter.

We analyze the complexity of the PV algorithm in its two different versions, roughly speaking, one version that corresponds to the arithmetic complexity and the second that corresponds to the (arguably more realistic) bit complexity. We should note that our analysis contains the subdivision routine of the PV algorithm for curves and surfaces as the special cases for $n = 2$ and $n = 3$, where we aim for estimates that holds for any n. We conduct average and smoothed analysis of the two versions of the PV algorithm, so we provide four different complexity analysis.

Average analysis framework is perhaps familiar to all of our readers, but the smoothed analysis might require a bit of an explanation: Suppose we endow the space of n-variate degree d polynomials with a norm $\|.\|$, and a probability measure μ (with as few assumptions as possible on μ). Suppose g is a random polynomial distributed with respect to μ. Then we consider an arbitrary polynomial f, and we fix a tolerance parameter $\sigma > 0$. We consider $q = f + \sigma\|f\|g$ as random perturbation of f with tolerance σ, and conduct average analysis of the PV algorithm for q. This type of an estimate could a priori depend on the arbitrary polynomial f. We aim for a uniform estimate that provides an upper bound for any f, and depends only on σ, n, and d. This uniform upper bound will be the smoothed analysis of the PV algorithm. It turns out that this random perturbation idea was already considered in computational geometry literature in an experimental fashion, and there were aims for building a theoretical framework (see section 4 of [22]).

Our main results Theorem 3.1 and Theorem 3.2 provide the four promised estimates on the complexity of PV algorithm for any number of variables n. For the special case of the plane curves, the average and smoothed analysis of the arithmetic complexity of the PV algorithm are respectively $\mathcal{O}(d^7)$ and $\mathcal{O}\left(d^7(1+\frac{1}{\sigma})^3\right)$. The average and smoothed analysis of the bit complexity are just slightly worse: $\mathcal{O}(d^7\log^2 d)$ and $\mathcal{O}(d^7\log^2 d\ (1+\frac{1}{\sigma})^3)$, respectively. These bounds are in marked contrast with the $\mathcal{O}(2^{\tau d^4\log d})$ *worst-case* complexity bound in [10].

For a clear presentation of our contribution and surrounding complexity considerations we need to make a few remarks:

(1) The use of floating-point arithmetic generates numerical errors which accumulate during the computation. An important remark is that, despite this accumulation of errors, our algorithm returns a correct output, a subdivision with the properties we want. It is, in this sense, a *certified* algorithm. At the heart of this remark is the fact that a sufficiently small perturbation of a correct subdivision is still a correct subdivision for a generic (i.e. non-singular) input. Condition numbers allow us to estimate how large this perturbation may be. Then, the fact that we can estimate these condition numbers, we control the precision of the operations' round-off, and we know how these operations are sequenced further allows us to ensure that the subdivision we constructed is close enough to the one we would have done in an error-free context.

Needless to say, for input data outside the set satisfying the generic property above our reasoning does not hold. The set of such inputs, referred to as *ill-posed* in numerical analysis, has measure zero. Condition numbers relate to ill-posedness in the sense that the closer a data is to the set of ill-posed inputs the larger becomes its condition number. It is these facts that allows one to establish average and smoothed analysis by means of probabilistic estimates on the condition numbers. This general scheme was proposed in [32]. A more detailed discussion of these issues is in [4, §9.5]. A relatively early case of a fully studied variable-precision algorithm is in [12]. An account of the use of floating-point arithmetic in computational geometry is given in [22].

(2) Most of the probabilistic analyses for cost measures or condition numbers use the Gaussian measure. This choice is mainly for technical convenience. For the analysis of condition numbers, this goes back to Goldstine and von Neumann [24] and, more recently, resulted in simple bounds for a large class of condition numbers [19, 1, 2, 27].

In the last few years, however, the search for more robust complexity analysis resulted in estimates that hold for a (quite) general family of measures. The family of *subgaussian* measures which includes all compactly supported random variables provides a good testing ground. An analysis of a condition number for these distributions occupies [21]. It is for this class of distributions (subgaussians with an anti-concentration property) that our results are proved.

(3) The subdivision procedure we analyze can be considered at three levels of generality: the *abstract*, in which we only take into account the number of iterations of the subdivision procedure; the *interval*, in which we take also into account the number of arithmetic operations; and the *effective*, in which we take into account not only the number of arithmetic operations, but also the precision that they need, obtaining a realistic estimation of the bit-cost of the algorithm. This division follows a trend for analysing subdivision algorithms initiated by Xu and Yap [36] (cf. [37]).

Our condition-based analysis can be applied at each of these three levels, hopefully showing the usefulness of the approach. Whereas this paper focuses on a particular subdivision procedure we believe that the techniques in this paper can be readily applied to other subdivision based algorithms in computational geometry. We note, however, that the complexity analysis in this paper would have been impossible without the *continuous amortization* technique developed in the exact numerical context [8, 9]. In this regard, we hope to trigger a fruitful exchange of ideas between the different approaches to continuous computation and improve our (seemingly preliminary) understanding of the complexity of subdivision algorithms in computational geometry.

Outline

The rest of the paper is structured as follows: We start with a section that contains notation. We beg readers' pardon for this inconvenient start; this seemed the simplest way for getting things clear. Then in Section 2 we discuss the Plantinga-Vegter algorithm and the n-dimensional generalization of its subdivision method in the abstract, the interval arithmetic, and the effective versions. Section 3 introduces our randomness model and contains main complexity estimates of this paper. In Section 4, we present a geometric framework (read Hilbert space structure) to deal with homogeneous polynomials. In Section 5, we introduce the condition number κ_{aff} —both local, i.e., at a point x, and global— along with its main properties. In Section 6, we present the existing results on the complexity of Plantinga-Vegter algorithm from [10], and we relate these results to the local condition number. In Section 7, we carry out the finite-precision analysis deriving the corresponding bounds for bit-cost. Finally, in Section 8, we derive average and smoothed complexity bounds under (quite) general randomness assumptions.

Notation

Throughout the paper, we will assume some familiarity with the basics of differential geometry and with the sphere $\mathbb{S}^n$ as a Riemannian manifold. For scalar smooth maps $f : \mathbb{R}^m \to \mathbb{R}$, we will write the tangent map at $x \in \mathbb{R}^m$ as $\partial_x f : \mathbb{R}^m \to \mathbb{R}$ when we want to emphasize it as a linear map and as $\partial f : \mathbb{R}^m \to \mathbb{R}^m$, $x \mapsto \partial f(x)$, when we want to emphasize it as a smooth function. For general smooth maps between smooth manifolds $F : \mathcal{M} \to \mathcal{N}$, we will just write $\partial_x F : \mathrm{T}_x\mathcal{M} \to \mathrm{T}_x\mathcal{N}$ as the tangent map.

In what follows, $\mathcal{P}_{n,d}$ will denote the set of real polynomials in the n variables $X_1, \dots, X_n$ with degree at most d, $\mathcal{H}_{n,d}$ the set of homogeneous real polynomials in the $n+1$ variables $X_0, X_1, \dots, X_n$ of degree d, and $\|\ \|$ and $\langle\ ,\ \rangle$ will denote the standard norm and inner product in $\mathbb{R}^m$ as well as the Weyl norm and inner product in $\mathcal{P}^m_{n,d}$ and $\mathcal{H}^m_{n,d}$. Given a polynomial $f \in \mathcal{P}_{n,d}$, $f^{\mathsf{h}} \in \mathcal{H}_{n,d}$ will be its homogenization and ∂f the polynomial map given by its partial derivatives. We will denote by the Cyrillic character Ю, 'yu', the central projection (4.1) that maps $\mathbb{R}^n$ into $\mathbb{S}^n$. For details see Section 4. Additionally, $V_{\mathbb{R}}(f)$ and $V_{\mathbb{C}}(f)$ will be, respectively, the real and complex zero sets of f.

We will denote by $\square X$ the set of n-boxes of the form $x + I^n$, where I is an interval, that are contained in X and, for a given box $B \in \square\mathbb{R}^n$, $m(B)$ will be its middle point, $w(B)$ its width, and $\operatorname{vol} B = w(B)^n$ its volume.

Regarding probabilistic conventions, we will denote the probability of an event by $\mathbb{P}$, random variables by $\mathfrak{x}, \mathfrak{y}, \dots$ and random polynomials by $\mathfrak{f}, \mathfrak{g}, \mathfrak{q}, \dots$ The expression $\mathbb{E}_{\mathfrak{x}\in K} g(\mathfrak{x})$ will denote the expectation of $g(\mathfrak{x})$ when $\mathfrak{x}$ is sampled uniformly from the set K and $\mathbb{E}_{\mathfrak{y}} g(\mathfrak{y})$ the expectation of $g(\mathfrak{y})$ with respect to a previously specified probability distribution of $\mathfrak{y}$.

Regarding complexity parameters, n will be the number of variables, d the degree bound, and $N = \binom{n+d}{n}$ the dimension of $\mathcal{P}_{n,d}$. Finally, ln will denote the natural logarithm and log the logarithm in base 2.

2 The Plantinga-Vegter (Subdivision) Algorithm

Given a real smooth hypersurface in $\mathbb{R}^n$ described implicitly by a map $f : \mathbb{R}^n \to \mathbb{R}$ and a region $[-a,a]^n$, the Plantinga-Vegter Algorithm constructs a piecewise-linear approximation of the intersection of its zero set $V_{\mathbb{R}}(f)$ with $[-a,a]^n$ isotopic to this intersection inside $[-a,a]^n$. The Plantinga-Vegter algorithm (see Figure 2.1 for an illustration) is divided in two phases:

1) Subdivision phase: In this phase, the Plantinga-Vegter algorithm subdivides $[-a,a]^n$ into smaller and smaller boxes until all the boxes satisfy a certain condition (see (2.1)).
2) Post-processing phase: In this phase, the Plantinga-Vegter algorithm uses the obtained subdivision to produce a piecewise-linear approximation of the given hypersurface.

We will focus on the subdivision phase of the Plantinga-Vegter algorithm. We do this because the complexity of subdivision-based algorithms is usually dominated by the complexity of the subdivision phase. This follows the guidelines of the first complexity analysis given by Burr, Gao and Tsigaridas [10] (cf. [11]).

We note that it would be interesting to incorporate the complexity of the post-processing phase of the algorithm to our estimates in this paper: either the original one by Plantinga-Vegter [28], for $n \leq 3$, or the generalization to higher dimensions by Galehouse [23], for arbitrary n. We also don't cover existing extensions of the Plantinga-Vegter algorithm to singular curves [7].

From now on, when we say Plantinga-Vegter algorithm, we are referring to the Plantinga-Vegter subdivision phase, and we restrict to the case in which $f : \mathbb{R}^n \to \mathbb{R}$ is a polynomial. We now describe this algorithm at the three levels: abstract, interval and effective.

2.1 Abstract level: Algorithm PV-Abstract

The Plantinga-Vegter algorithm subdivides $[-a,a]^n$ until a certain regularity condition is satisfied in each of the boxes B of the subdivision. Let $h, h' : \mathbb{R}^n \to (0,\infty)$ be some fixed positive maps, conveniently chosen (see (2.3) below). Then this regularity condition is

$$C_f(B)\text{: either } 0 \notin (hf)(B) \text{ or } 0 \notin \langle (h'\partial f)(B), (h'\partial f)(B)\rangle. \tag{2.1}$$

Note that this condition is satisfied when either B does not contain any zero of f or no pair of gradient vectors of f are orthogonal in B. Intuitively, the latter means that the level sets of f should approximately look like a set of parallel hyperplanes.

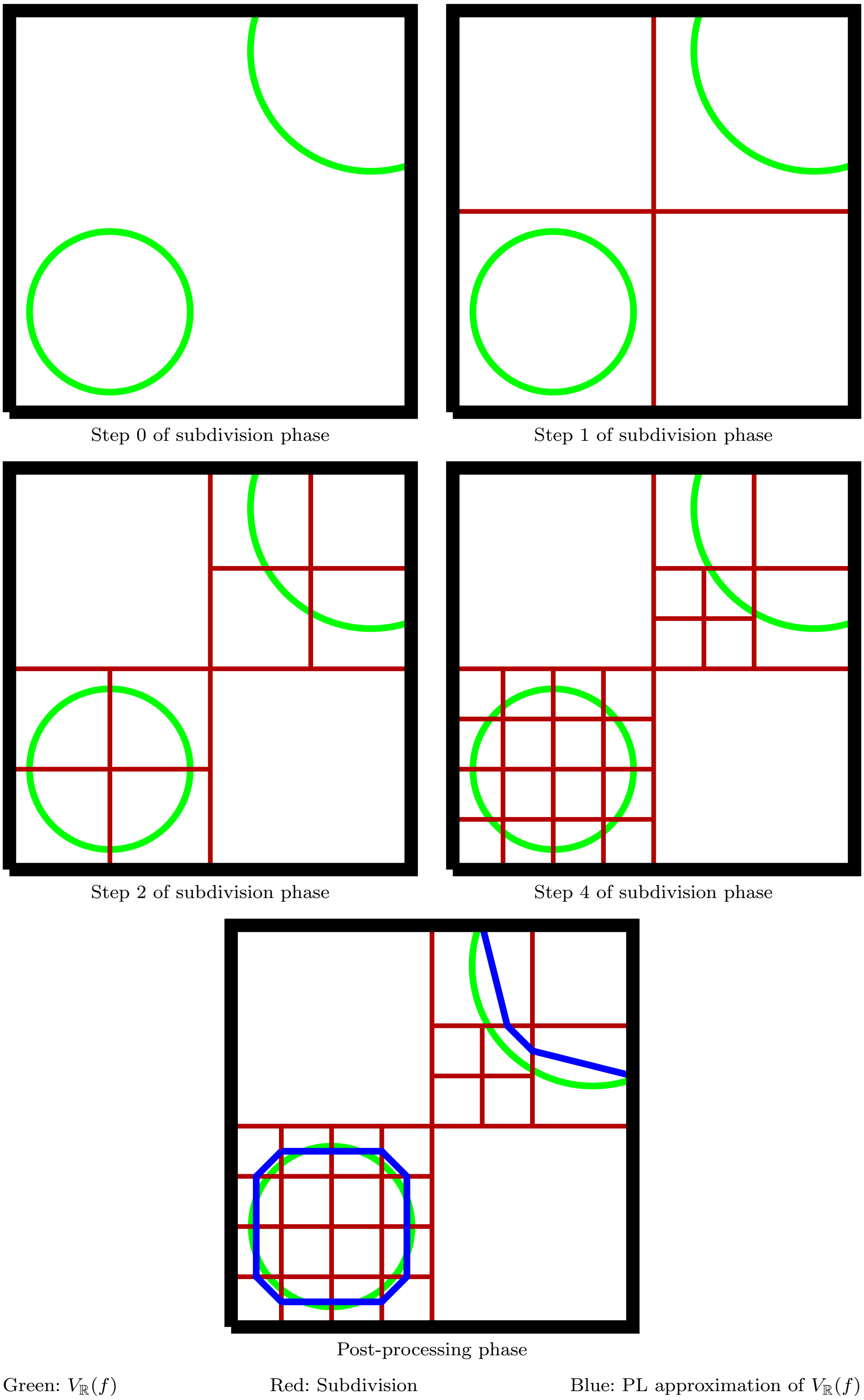

Fig. 2.1 Plantinga-Vegter applied to $f = X^4 - 6X^3 + 2X^2Y^2 - 6X^2Y - 34X^2 - 6XY^2 - 320XY + 376X + Y^4 - 6Y^3 - 34Y^2 + 376Y + 3128$ in $[-10, 10]^2$. [34, Figure 5§2 1]

In its abstract form, the Plantinga-Vegter algorithm is described in Algorithm PV-Abstract below. The StandardSubdivision procedure in the description refers to taking a box B and subdividing it into 2^n boxes of equal size.

Algorithm 1: PV-Abstract

Input : $f : \mathbb{R}^n \to \mathbb{R}$ with interval approximations $\square[hf]$ and $\square[h'\nabla f]$
$a \in (0, \infty)$
Precondition : $V_{\mathbb{R}}(f)$ is smooth inside $[-a, a]^n$

$\tilde{\mathcal{S}} \leftarrow \{[-a, a]^n\}$
$\mathcal{S} \leftarrow \varnothing$
repeat
  Take B in $\tilde{\mathcal{S}}$
  $\tilde{\mathcal{S}} \leftarrow \tilde{\mathcal{S}} \setminus \{B\}$
  if $C_f(B)$ `true` **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{B\}$
  else
    $\tilde{\mathcal{S}} \leftarrow \tilde{\mathcal{S}} \cup$ StandardSubdivision(B)
until $\tilde{\mathcal{S}} = \varnothing$
return $\mathcal{S}$

Output : Subdivision $\mathcal{S} \subseteq \square[-a, a]^n$ of $[-a, a]^n$
Postcondition : For all $B \in \mathcal{S}$, $C_f(B)$ is true

2.2 Interval level: Algorithm PV-Interval

To check condition $C_f(B)$, we use interval approximations allowing us to certify whether or not 0 is in the image of B under a certain map. Recall that an *interval approximation* [29] of a function $F : \mathbb{R}^m \to \mathbb{R}^{m'}$ is a map

$$\square[F] : \square\mathbb{R}^m \to \square\mathbb{R}^{m'}$$

such that for all $B \in \square\mathbb{R}^m$,

$$F(B) \subseteq \square[F](B). \tag{2.2}$$

We notice that if we see B as an error bound for its midpoint $m(B)$, then we can see $\square[F](B)$ as an error bound for $F(m(B))$.

A natural choice for the interval approximation of a C^1-function $F : \mathbb{R}^m \to \mathbb{R}^{m'}$ is its *standard interval approximation*

$$\square\mathbb{R}^m \ni B \mapsto \square_{\mathrm{std}}[F](B) := F(m(B)) + \sqrt{m}\left(\sup_{x\in B} \|\mathrm{D}_x F\|\right)\left[-\frac{w(B)}{2}, \frac{w(B)}{2}\right]^{m'}$$

where $\mathrm{D}_x F$ is the tangent map of F at x and $\|\mathrm{D}_x F\|$ its operator norm. Note that to construct this one in practice, we need to be able to evaluate F and to compute efficiently upper bounds for $\sup_{x\in B} \|\mathrm{D}_x F\|$. In our case, this is possible due to the fact that we are working with polynomials.

Let $f \in \mathcal{P}_{n,d}$. We will consider

$$h(x) = \frac{1}{\|f\|(1 + \|x\|^2)^{(d-1)/2}} \quad \text{and} \quad h'(x) = \frac{1}{d\|f\|(1 + \|x\|^2)^{d/2-1}} \tag{2.3}$$

along with the maps

$$\widehat{f} : x \mapsto h(x)f(x) = \frac{f(x)}{\|f\|(1 + \|x\|^2)^{(d-1)/2}} \tag{2.4}$$

and

$$\widehat{\partial f} : x \mapsto h'(x)\partial f(x) = \frac{\partial f(x)}{d\|f\|(1 + \|x\|^2)^{d/2-1}} \tag{2.5}$$

where $\|f\|$ is the Weyl norm of f (which we recall in Definition 4.2). These are "linearized" versions of f and its derivative. The intuition behind this fact is that for large values of x a polynomial map of degree d grows like $\|x\|^d$. This Lipschitz property allows us to prove the following (in Subsection 4.3).

Proposition 2.1 *Let $f \in \mathcal{P}_{n,d}$. Then*

$$\square[hf] : B \mapsto \widehat{f}(m(B)) + (1+\sqrt{d})\sqrt{n}\left[-\frac{w(B)}{2}, \frac{w(B)}{2}\right]$$

is an interval approximation of hf, and

$$\square[h'\partial f] : B \mapsto \widehat{\partial f}(m(B)) + \left(1+\sqrt{d-1}\right)\sqrt{n}\left[-\frac{w(B)}{2}, \frac{w(B)}{2}\right]^n$$

is an interval approximation of $h'\partial f$.

Moreover, we note that checking the condition "$0 \notin \langle B, B\rangle$" for a box B can be reduced to checking

$$\sqrt{\frac{n}{2}}w(B) \leq \|m(B)\|.$$

To do the latter we will use Lemma 4.2 (which we also prove in Subsection 4.3). Together with the interval approximations in Proposition 2.1, we derive a condition $C_f^{\square}$, implying $C_f(B)$ and easy to check.

Theorem 2.1 *Let $B \in \square\mathbb{R}^n$. If the condition*

$$C_f^{\square}(B)\ :\ \left|\widehat{f}(m(B))\right| > 2\sqrt{dn}w(B) \quad or \quad \left\|\widehat{\partial f}(m(B))\right\| > 2\sqrt{2}\sqrt{dn}w(B).$$

is satisfied, then $C_f(B)$ is true.

Theorem 2.1 is the basis of the interval version of Algorithm PV-Interval below.

Algorithm 2: PV-Interval

Input : $f \in \mathcal{P}_{n,d}$
$a \in (0, \infty)$
Precondition : $V_{\mathbb{R}}(f)$ is smooth inside $[-a,a]^n$

$\tilde{\mathcal{S}} \leftarrow \{[-a,a]^n\}$
$\mathcal{S} \leftarrow \varnothing$
repeat
  Take B in $\tilde{\mathcal{S}}$
  $\tilde{\mathcal{S}} \leftarrow \tilde{\mathcal{S}} \setminus \{B\}$
  if $\left|\widehat{f}(m(B))\right| > (1+\sqrt{d})\sqrt{n}w(B)$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{B\}$
  else if $\left\|\widehat{\partial f}(m(B))\right\| > \sqrt{2}(1+\sqrt{d-1})nw(B)$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{B\}$
  else
    $\tilde{\mathcal{S}} \leftarrow \tilde{\mathcal{S}} \cup \textsc{StandardSubdivision}(B)$
until $\tilde{\mathcal{S}} = \varnothing$
return $\mathcal{S}$

Output : Subdivision $\mathcal{S} \subseteq \square[-a,a]^n$ of $[-a,a]^n$
Postcondition : For all $B \in \mathcal{S}$, $C_f(B)$ is true

Remark 2.1 We could consider alternative interval approximations. For instance, the interval approximations in [10], which we will refer to as BGT, are based on the Taylor expansion at the midpoint, so they differ from the ones in Proposition 2.1. In the interlude at the end of Section 6, we will show that our complexity analysis also applies to this interval approximation.

2.3 Effective level: Algorithm PV-Effective

For the effective version (Algorithm PV-Effective), we will use floating-point numbers (cf. [4, §O.3.1] or [25, §1.2]). We do this, instead of using fixed-point or big rationals, because the use of floating-point is computationally cheap, both in time and space.

A floating-point number has the form

$$\pm\, 0.a_1 a_2 \cdots a_{\mathbf{m}}\, 2^e$$

where $a_1, \dots, a_m \in \{0,1\}$ and $e \in \mathbb{Z}$. In general, the *number of significant digits*, $\mathbf{m}$, is fixed during the computation of arithmetic expressions, but it can be updated at different iterations of an algorithm if an increase in precision is needed.

We note that every real number $x \in \mathbb{R}$ has a floating-point approximation $r_{\mathbf{m}}(x)$ with $\mathbf{m}$ digits, such that

$$r_{\mathbf{m}}(x) = x(1+\delta)$$

for some $\delta \in (-2^{-(\mathbf{m}-1)}, 2^{-(\mathbf{m}-1)})$. Moreover, given two floating-point numbers x and y with $\mathbf{m}$ significant digits, we can easily compute

$$r_{\mathbf{m}}(x+y),\ r_{\mathbf{m}}(x-y),\ r_{\mathbf{m}}(xy),\ r_{\mathbf{m}}(x/y), \text{ and } r_{\mathbf{m}}\left(\sqrt{x}\right)$$

in $\mathcal{O}(\mathbf{m}^2)$ bit-operations. Comparisons between floating-point numbers can also be made using this amount of bit-operations.

Remark 2.2 In the above estimation we are ignoring the complexity of adding the exponents or operating with them. In general the size of e is of the order of $|\log|x||$, and so the bit-size of e is of the order of $|\log|\log|x|||$. This means that, unless the numbers we deal with are enormous, one should not worry about the bit-size of e for cost estimates.

Finite-precision analyses do not rely on the precise form of floating-point numbers but just in some general properties which we now summarize. There is a subset $\mathbb{F} \subset \mathbb{R}$ of *floating-point numbers* (which we assume contains 0), a *rounding map* $r : \mathbb{R} \to \mathbb{F}$, and a *round-off unit* $\mathbf{u} \in (0,1)$ satisfying the following conditions:

(i) For any $x \in \mathbb{F}$, $r(x) = x$. In particular, $r(0) = 0$.
(ii) For any $x \in \mathbb{R}$, $r(x) = x(1+\delta)$ with $|\delta| \leq \mathbf{u}$.

Moreover, for $\circ \in \{+, -, \times, /\}$, there are approximate versions

$$\tilde{\circ} : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$$

such that for all $x, y \in \mathbb{F}$,

$$x \,\tilde{\circ}\, y = (x \circ y)(1+\delta) \tag{2.6}$$

for some δ such that $|\delta| < \mathbf{u}$. We also assume that there is

$$\widetilde{\sqrt{\ }} : \mathbb{F} \to \mathbb{F}$$

such that for all $x \in \mathbb{F}$ with $x \geq 0$,

$$\widetilde{\sqrt{x}} = \sqrt{x}(1+\delta)$$

for some δ such that $|\delta| < \mathbf{u}$. Each of these operations and comparisons between numbers in $\mathbb{F}$ can be done with cost $\mathcal{O}\left(\log^2 \frac{1}{\mathbf{u}}\right)$. For the floating-point numbers we described above we have $\mathbf{u} = 2^{-(\mathbf{m}-1)}$.

Once the way we deal with finite precision is clear, we introduce the efficient version of the Plantinga-Vegter algorithm (PV-Effective below). We note that the algorithm updates the number of significant digits, $\mathbf{m} := |\log \mathbf{u}| + 1$, depending on the width of the box that is being considered, being able, if necessary, to read the coefficients of f with this updated precision.

Algorithm 3: PV-EFFECTIVE

Input: $f \in \mathcal{P}_{n,d}$
$a \in [1, \infty)$
Precondition : $V_{\mathbb{R}}(f)$ is smooth inside $[-a,a]^n$

$\mathbf{m}_0 \leftarrow 7 + \left\lceil \log \sqrt{dn} \right\rceil$
$\tilde{\mathcal{S}} \leftarrow \{[-a,a]^n\}$
$\mathcal{S} \leftarrow \varnothing$
repeat
  Take B in $\tilde{\mathcal{S}}$
  $\tilde{\mathcal{S}} \leftarrow \tilde{\mathcal{S}} \setminus \{B\}$
  $\mathbf{m}_B \leftarrow \mathbf{m}_0 + \lceil \max\{\log a, \log(a/w(B))\} \rceil$
  Switch to floating-point numbers with $\mathbf{m}_B$ significant digits
  if $\left|\widehat{f}(m(B))\right| > 4\sqrt{dn}w(B)$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{B\}$
  else if $\left\|\widehat{\partial f}(m(B))\right\| > 6\sqrt{d}nw(B)$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup \{B\}$
  else
    $\tilde{\mathcal{S}} \leftarrow \tilde{\mathcal{S}} \cup$ STANDARDSUBDIVISION(B)
until $\tilde{\mathcal{S}} = \varnothing$
return $\mathcal{S}$

Output: Subdivision $\mathcal{S} \subseteq \square[-a,a]^n$ of $[-a,a]^n$
Postcondition : For all $B \in \mathcal{S}$, $C_f(B)$ is true

3 Main results

In this section, we outline without proofs the main results of this paper. In the first part, we describe our randomness assumptions for polynomials. In the second one, we give precise statements for our bounds on the average and smoothed complexity of phase I of the Plantinga-Vegter Algorithm with infinite precision. In the last part, we state similar results in the context of finite-precision arithmetic.

3.1 Randomness Model

Most of the literature on random multivariate polynomials considers polynomials with Gaussian independent coefficients and relies on techniques that are only useful for Gaussian measures. We will instead consider a general family of measures relying on robust techniques coming from geometric functional analysis. Let us recall some basic definitions.

(P1) A random variable $\mathfrak{x} \in \mathbb{R}$ is called *centered* if $\mathbb{E}\mathfrak{x} = 0$.
(P2) A random variable $\mathfrak{x} \in \mathbb{R}$ is called *subgaussian* if there exists a K such that for all $p \geq 1$,

$$(\mathbb{E}|\mathfrak{x}|^p)^{\frac{1}{p}} \leq K\sqrt{p}.$$

The smallest such K is called the *Ψ_2-norm* of $\mathfrak{x}$.
(P3) A random variable $\mathfrak{x} \in \mathbb{R}$ satisfies the *anti-concentration property with constant ρ* if

$$\max\{\mathbb{P}(|\mathfrak{x} - u| \leq \varepsilon) \mid u \in \mathbb{R}\} \leq \rho\varepsilon.$$

The subgaussian property (P2) has other equivalent formulations. We refer the interested reader to [35]. We note that the anti-concentration property (P3) is equivalent to having a density (with respect to the Lebesgue measure) bounded by $\rho/2$.

Definition 3.1 A *dobro random polynomial* $\mathfrak{f} \in \mathcal{H}_{n,d}$ with parameters K and ρ is a polynomial

$$\mathfrak{f} := \sum_{|\alpha|=d} \binom{d}{\alpha}^{\frac{1}{2}} \mathfrak{c}_\alpha X^\alpha \tag{3.1}$$

such that the $\mathfrak{c}_\alpha$ are independent centered subgaussian random variables with Ψ_2-norm at most K and anti-concentration property with constant ρ. A *dobro random polynomial* $\mathfrak{f} \in \mathcal{P}_{n,d}$ is a polynomial f such that its homogenization $\mathfrak{f}^{\mathsf{h}}$ is so.

Remark 3.1 The word "dobro" comes from Russian and it means good. The word "dobra" in Turkish means straight and honest, and the word has similar connotations in Greek.

Some dobro random polynomials of interest are the following three.

(N) A *KSS random polynomial* is a dobro random polynomial such that each $\mathfrak{c}_\alpha$ in (3.1) is Gaussian with unit variance. For this model we have $K\rho = 1/\sqrt{2\pi}$.

(U) A *Weyl random polynomial* is a dobro random polynomial such that each $\mathfrak{c}_\alpha$ in (3.1) have uniform distribution in $[-1, 1]$. For this model we have $K\rho \leq 1$.

(E) For $\ell \geq 2$, a *ℓ-random polynomial* is a dobro random polynomial whose coefficients are independent identically distributed random variables with density function

$$t \mapsto \frac{1}{2\Gamma\left(1+\frac{1}{\ell}\right)}\, e^{-|t|^\ell}.$$

We have in this case that $\rho \leq 1$ and $K \leq 6/5$.

Remark 3.2 The relevant complexity parameter for a dobro random polynomial $\mathfrak{f} \in \mathcal{P}_{n,d}$ with constants K and ρ is the product $K\rho$. This is so because this product is invariant under scalings of $\mathfrak{f}$ and condition numbers will be scale-invariant. Note that $t\mathfrak{f}$ is still dobro, but with constants tK and ρ/t.

Remark 3.3 If we are interested in integer polynomials, dobro random polynomials may seem inadequate. One may be inclined to consider random polynomials $\mathfrak{f} \in \mathcal{P}_{n,d}$ such that $\mathfrak{c}_\alpha$ is a random integer in the interval $[-2^\tau, 2^\tau]$, i.e., $\mathfrak{c}_\alpha$ is a random integer of bit-size at most τ. As $\tau \to \infty$ and after we normalize the coefficients dividing by 2^τ, this random model converges to that of Weyl random polynomials.

Yet, in order to have a more satisfactory understanding of random integer polynomials, one has to consider random variables without a continuous density function. The techniques used in this note have already been extended to deal with such distributions in the case of random matrices [30,35]. We hope to pursue this delicate case in a more general setting (including complete intersections) in future work.

3.2 Complexity at the interval and effective levels

The following three theorems give bounds for, respectively, the average and smoothed complexity of PV-Interval and PV-Effective.

Theorem 3.1 Complexity of PV-Interval*:*

(A) Let $\mathfrak{f} \in \mathcal{P}_{n,d}$ be a dobro random polynomial with parameters K and ρ. The expected number of boxes in the final subdivision $\mathcal{S}$ of PV-Interval *on input $(\mathfrak{f}, a)$ is at most*

$$d^n N^{\frac{n+1}{2}} \max\{1, a^n\} 2^{12n\log n+8} (K\rho)^{n+1}$$

and the expected number of arithmetic operations is at most

$$\mathcal{O}\left(d^{n+1} N^{\frac{n+3}{2}} \max\{1, a^n\} 2^{12n\log n+8} (K\rho)^{n+1}\right).$$

(S) Let $f \in \mathcal{P}_{n,d}$, $\sigma > 0$, and $\mathfrak{g} \in \mathcal{P}_{n,d}$ a dobro random polynomial with parameters $K \geq 1$ and ρ . Then the expected number of n-cubes of the final subdivision $\mathcal{S}$ of PV-Interval *on input $(\mathfrak{q}_\sigma, a)$ where $\mathfrak{q}_\sigma = f + \sigma\|f\|\mathfrak{g}$ is at most*

$$d^n N^{\frac{n+1}{2}} \max\{1, a^n\} 2^{12n\log n+8} (K\rho)^{n+1} \left(1+\frac{1}{\sigma}\right)^{n+1}$$

and the expected number of arithmetic operations is at most

$$\mathcal{O}\left(d^{n+1} N^{\frac{n+3}{2}} \max\{1, a^n\} 2^{12n\log n+8} (K\rho)^{n+1} \left(1+\frac{1}{\sigma}\right)^{n+1}\right).$$

Theorem 3.2 Complexity of PV-Effective*:*

(A) Let $\mathfrak{f} \in \mathcal{P}_{n,d}$ be a dobro random polynomial with parameters K and ρ. The expected number of boxes in the final subdivision $\mathcal{S}$ of PV-Effective *on input $(\mathfrak{f}, a)$ is at most*

$$d^n N^{\frac{n+1}{2}} a^n 2^{15n\log n+12}(K\rho)^{n+1}$$

and the expected number of arithmetic operations is at most

$$\mathcal{O}\left(d^{n+1} N^{\frac{n+3}{2}} a^n 2^{15n\log n+12}(K\rho)^{n+1}\right).$$

Moreover, the expected bit-cost of PV-Effective *on input $(\mathfrak{f}, a)$ is at most*

$$\mathcal{O}\left(d^{n+1} N^{\frac{n+3}{2}} a^n 2^{15n\log n+12}\log^2(dna)(K\rho)^{n+1}\right),$$

under the assumptions that floating-point arithmetic is done using standard arithmetic and that that the cost of operating with the exponents is negligible.

(S) Let $f \in \mathcal{P}_{n,d}$, $\sigma > 0$, and $\mathfrak{g} \in \mathcal{P}_{n,d}$ a dobro random polynomial with parameters $K \geq 1$ and ρ . Then the expected number of n-cubes of the final subdivision $\mathcal{S}$ of PV-Effective *on input $(\mathfrak{q}_\sigma, a)$ where $\mathfrak{q}_\sigma = f + \sigma\|f\|\mathfrak{g}$ is at most*

$$d^n N^{\frac{n+1}{2}} a^n 2^{15n\log n+12}(K\rho)^{n+1}\left(1+\frac{1}{\sigma}\right)^{n+1}$$

and the expected number of arithmetic operations is at most

$$\mathcal{O}\left(d^{n+1} N^{\frac{n+3}{2}} a^n 2^{15n\log n+12}(K\rho)^{n+1}\left(1+\frac{1}{\sigma}\right)^{n+1}\right).$$

Moreover, the expected bit-cost of PV-Effective *on input $(\mathfrak{q}_\sigma, a)$ is at most*

$$\mathcal{O}\left(d^{n+1} N^{\frac{n+3}{2}} a^n 2^{15n\log n+12}\log^2(dna)(K\rho)^{n+1}\left(1+\frac{1}{\sigma}\right)^{n+1}\right),$$

under the assumptions that floating-point arithmetic is done using standard arithmetic and that that the cost of operating with the exponents is negligible.

If n is fixed and d is let to vary, $N = \binom{n+d}{n} \leq e^n(1+\frac{d}{n})^n$. Hence the bounds of Theorems 3.1 and 3.2 are of the order $d^{\frac{n^2+5n}{2}}$ for any fixed n. The complexity estimate in [10, Theorem 4.3] reads as follows:

$$2^{\mathcal{O}\left(d^{n+1}(n\tau+nd\log(nd))n\log a\right)}$$

with τ being the largest bit-size of the coefficients of f. One can see that the smoothed analysis estimates are exponentially smaller than the worst case estimates, this seems to relate better with the practical efficiency of the Plantinga-Vegter algorithm.

We note, however, that the bound in [10] and our bounds cannot be directly compared. Not only because the former is worst-case and the latter average-case (or smoothed) but because of the different underlying settings: the bound in [10] applies to integer data, ours to real data. Nevertheless, the bounds for the effective version PV-Effective apply to the real data under finite precision and provides estimates for the bit complexity.

4 Geometric framework

There is an extensive literature on norms of polynomials and their relation to norms of gradients in $\mathcal{H}_{n,d}$. The PV algorithm, however, works in the affine space with non-homogenous polynomials. We first establish basic definitions and inequalities that allow us to translate existing results into the setting of the PV algorithm. After the transfer is completed, we continue with establishing interval approximations.

4.1 Weyl norm

We introduce the Weyl norm on $\mathcal{H}_{n,d}$.

Definition 4.1 The *Weyl norm* on $\mathcal{H}_{n,d}$ is the norm given by

$$\|f\| := \sqrt{\sum_\alpha \binom{d}{\alpha}^{-1} f_\alpha^2}$$

for $f = \sum_\alpha f_\alpha X^\alpha \in \mathcal{H}_{n,d}$; and the *Weyl norm* on $\mathcal{H}_{n,d}^q$ is the norm given by

$$\|\mathbf{f}\| := \sqrt{\sum_{i=1}^q \|f_i\|^2} = \sqrt{\sum_{i=1}^q \sum_\alpha \binom{d}{\alpha}^{-1} f_{i,\alpha}^2}$$

for $\mathbf{f} = (f_i) = (\sum_\alpha f_{i,\alpha} X^\alpha) \in \mathcal{H}_{n,d}^q$.

To extend this norm to $\mathcal{P}_{n,d}$, we use the homogeneization map

$$\begin{aligned} {}^{\mathsf{h}} : \mathcal{P}_{n,d} &\to \mathcal{H}_{n,d} \\ f &\mapsto f^{\mathsf{h}} := f(X_1/X_0, \dots, X_n/X_0)X_0^d. \end{aligned}$$

and its componentwise extension ${}^{\mathsf{h}} : \mathcal{P}_{n,d}^q \to \mathcal{H}_{n,d}^q$.

Definition 4.2 The *Weyl norm* on $\mathcal{P}_{n,d}^q$ is the norm given by

$$\|\mathbf{f}\| := \|\mathbf{f}^{\mathsf{h}}\|$$

for $\mathbf{f} \in \mathcal{P}_{n,d}^q$.

We remark that the Weyl norm comes from the corresponding inner product, so this gives means to perform orthogonal projections in polynomials spaces.

Note that for $F \in \mathcal{H}_{n,d}^q$, we have that $\partial F(X) \in \mathcal{H}_{n,d-1}^{q(n+1)}$ and so we can talk about the Weyl norm of $\partial F(X)$. The following proposition comes in handy.

Proposition 4.1 *Let* $\mathbf{f} \in \mathcal{H}_{n,d}^q$ *and* $y \in \mathbb{S}^n$. *Then,* (1) $\|\mathbf{f}(y)\| \leq \|\mathbf{f}\|$, (2) $\left\|\partial_y \mathbf{f}_{|\mathrm{T}_y\mathbb{S}^n}\right\| \leq \sqrt{d}\|\mathbf{f}\|$, (3) $\|\partial \mathbf{f}(X)\| \leq d\|\mathbf{f}\|$.

Proof (1) is [4, Lemma 16.6], (2) the Exclusion Lemma [4, Lemma 19.22], and (3) by a direct computation, arguing as in the proof of [4, Lemma 16.46]. Alternatively, one can also see [34, $1^{\S 1}$] for a direct account of the proofs. □

4.2 Central projection and homogeneization

Let $\text{Ꞁ}\!\!\text{O} : \mathbb{R}^n \to \mathbb{S}^n$ be the map given by

$$\text{Ꞁ}\!\!\text{O} : x \mapsto \frac{1}{\sqrt{1+\|x\|^2}} \begin{pmatrix} 1 \\ x \end{pmatrix}. \tag{4.1}$$

One can see that $\text{Ꞁ}\!\!\text{O}$ is the map induce by the central projection of $\mathbb{R}^n \times \{1\}$ onto the sphere $\mathbb{S}^n$ and that this map induces a diffeomorphism between $\mathbb{R}^n$ and the upper half of $\mathbb{S}^n$.

Given $\mathbf{f} \in \mathcal{P}_{n,d}^q$, we observe that

$$\mathbf{f}^{\mathsf{h}}(\text{Ꞁ}\!\!\text{O}(x)) = \frac{\mathbf{f}(x)}{(1+\|x\|^2)^{d/2}}, \tag{4.2}$$

and so, by the chain rule,

$$\partial_{\text{Ꞁ}\!\!\text{O}(x)} \mathbf{f}^{\mathsf{h}} \partial_x \text{Ꞁ}\!\!\text{O} = \frac{\partial_x \mathbf{f}}{(1+\|x\|^2)^{d/2}} - \frac{d \cdot \mathbf{f}(x) x^{\mathrm{T}}}{(1+\|x\|^2)^{d/2+1}} \tag{4.3}$$

where $\partial_y \mathbf{f}^{\mathsf{h}} : \mathbb{R}^{n+1} \to \mathbb{R}$, $\partial_x Ꞁ\mathrm{O} : \mathbb{R}^n \to \mathrm{T}_x\mathbb{S}^n = x^\perp$ and $\partial_x \mathbf{f} : \mathbb{R}^n \to \mathbb{R}$ are respectively the tangent maps of $\mathbf{f}^{\mathsf{h}}$, $Ꞁ\mathrm{O}$ and $\mathbf{f}$.

It is important to note that $Ꞁ\mathrm{O}$ deforms the metric. For each $x \in \mathbb{R}^n$, we can see that the singular values of $\partial_x Ꞁ\mathrm{O}$ are

$$\sigma_1 (\partial_x Ꞁ\mathrm{O}) = \cdots = \sigma_{n-1} (\partial_x Ꞁ\mathrm{O}) = \frac{1}{\sqrt{1+\|x\|^2}},\ \sigma_n (\partial_x Ꞁ\mathrm{O}) = \frac{1}{1+\|x\|^2}, \tag{4.4}$$

and so, in particular,

$$\|\partial_x Ꞁ\mathrm{O}\| = \frac{1}{\sqrt{1+\|x\|^2}}. \tag{4.5}$$

With the above, we reprove a version of Proposition 4.1 for $\mathcal{P}^q_{n,d}$ instead.

Proposition 4.2 *Let $\mathbf{f} \in \mathcal{P}^k_{n,d}$ be a polynomial map. Then the map*

$$\mathbf{F} : x \mapsto \frac{\mathbf{f}(x)}{\|\mathbf{f}\|(1+\|x\|^2)^{(d-1)/2}}$$

is $(1+\sqrt{d})$-Lipschitz and, for all x, $\left\|\mathbf{F}(x)\right\| \leq \sqrt{1+\|x\|^2}$.

Proof For the Lipschitz property, it is enough to bound the norm of the derivative of the map by $1+\sqrt{d}$. Due to (4.2),

$$\mathbf{F}(x) = \sqrt{1+\|x\|^2}\,\frac{\mathbf{f}^{\mathsf{h}}(Ꞁ\mathrm{O}(x))}{\|\mathbf{f}\|}$$

and so, by the chain rule,

$$\partial_x \mathbf{F} = \frac{\mathbf{f}^{\mathsf{h}}(Ꞁ\mathrm{O}(x))}{\|\mathbf{f}\|}\frac{x^{\mathrm{T}}}{\sqrt{1+\|x\|^2}} + \sqrt{1+\|x\|^2}\,\frac{\partial_{\phi(x)}\mathbf{f}\,\partial_x Ꞁ\mathrm{O}}{\|\mathbf{f}\|}.$$

Now, by the triangle inequality,

$$\|\partial_x \mathbf{F}\| \leq \frac{\|\mathbf{f}^{\mathsf{h}}(Ꞁ\mathrm{O}(x))\|}{\|\mathbf{f}\|}\frac{\|x\|}{\sqrt{1+\|x\|^2}} + \sqrt{1+\|x\|^2}\,\frac{\|\partial_{\phi(x)}\mathbf{f}\,\partial_x Ꞁ\mathrm{O}\|}{\|\mathbf{f}\|}.$$

On the one hand,

$$\frac{\|\mathbf{f}^{\mathsf{h}}(Ꞁ\mathrm{O}(x))\|}{\|\mathbf{f}\|} \leq 1,$$

by Proposition 4.1 (1). On the other hand,

$$\|\partial_{\phi(x)}\mathbf{f}\,\partial_x Ꞁ\mathrm{O}\| = \left\|\partial_{Ꞁ\mathrm{O}(x)}\mathbf{f}_{|\mathrm{T}_{Ꞁ\mathrm{O}(x)}\mathbb{S}^n}\,\partial_x Ꞁ\mathrm{O}\right\| \leq \left\|\partial_{Ꞁ\mathrm{O}(x)}\mathbf{f}_{|\mathrm{T}_{Ꞁ\mathrm{O}(x)}\mathbb{S}^n}\right\|\,\|\partial_x Ꞁ\mathrm{O}\| \leq \frac{\sqrt{d}\|\mathbf{f}\|}{\sqrt{1+\|x\|^2}},$$

by Proposition 4.1 (2) and (4.5). Hence

$$\|\partial_x \mathbf{F}\| \leq \frac{\|x\|}{\sqrt{1+\|x\|^2}} + \sqrt{d} \leq 1+\sqrt{d}$$

as we wanted to show. The claim about $\left\|\mathbf{F}(x)\right\|$ follows from Proposition 4.1 (1) applied to the formula above. $\square$

4.3 Interval approximations

Recall that our interval approximations, given in Proposition 2.1, rely on the functions $\widehat{f}$ and $\widehat{\partial f}$ given, respectively, in (2.4) and (2.5). The following lemma will give us the justification of our interval approximations, and with it a proof of Proposition 2.1.

Lemma 4.1 *Let $f \in \mathcal{P}_{n,d}$. Then:*

(1) The map $\widehat{f}$ given in (2.4) *is $(1+\sqrt{d})$-Lipschitz and for all $x \in \mathbb{R}^n$, it satisfies $|\widehat{f}(x)| \leq \sqrt{1+\|x\|^2}$.*
(2) The map $\widehat{\partial f}$ given in (2.5) *is $(1+\sqrt{d-1})$-Lipschitz and for all $x \in \mathbb{R}^n$, it satisfies $\|\widehat{\partial f}(x)\| \leq \sqrt{1+\|x\|^2}$.*

Proof (Proof of Proposition 2.1) It is a straightforward consequence of the Lipschitz properties in Lemma 4.1. □

Proof (Proof of Lemma 4.1) (1) Apply Proposition 4.2 with $\mathbf{f} = f$, then $\widehat{f} = \mathbf{F}$ and the claim follows.

(2) Apply Proposition 4.2 with $\mathbf{f} = f$, then $\widehat{\partial f} = \frac{\|\partial f(X)\|}{d\|f\|}\mathbf{F}$ and the claims follows since $\frac{\|\partial f(X)\|}{d\|f\|} \leq 1$ by Proposition 4.1 (3). □

Once we have shown that our interval approximations are so, we show Theorem 2.1 which reduces the interval condition $C_f(B)$ to the condition $C'_f(B)$ at a point.

Lemma 4.2 *Let $x \in \mathbb{R}^n$ and $s \in [0, 1/\sqrt{2}]$. Then for all $v, w \in B(x, s\|x\|)$, we have $\langle v, w\rangle > \|v\|\|w\|(1-2s^2) \geq 0$.*

Proof (Proof of Theorem 2.1) By the ℓ_2-ℓ_∞ inequality, interval approximations of Proposition 2.1 satisfy that for all $B \in \square\mathbb{R}^n$

$$\operatorname{dist}\left((hf)(m(B)), \square[hf](B)\right) \leq \left(1+\sqrt{d}\right)\sqrt{n}\, w(B)/2 \tag{4.6}$$

and

$$\operatorname{dist}\left((h'\partial f)(m(J)), \square[h'\partial f](B)\right) \leq \left(1+\sqrt{d-1}\right) n w(B)/2 \tag{4.7}$$

where dist is the usual Euclidean distance.

When the condition on $\widehat{f}(m(J))$ is satisfied, (4.6) guarantees that $0 \notin \square[hf](J)$. Whenever the condition on $\widehat{\partial f}(m(J))$ is satisfied, (4.7) and Lemma 4.2 (with $s = 1/\sqrt{2}$) guarantee that $0 \notin \langle \square[h'\partial f](J), \square[h'\partial f](J)\rangle$. Hence $C'_f(B)$ implies $C_f(B)$. □

Proof (Proof of Lemma 4.2) Let $s = \cos\theta$, so that $\theta \in [0, \pi/4]$, $c = \sqrt{1-s^2}$ and $K_c := \{u \in \mathbb{R}^n \mid \langle x, u\rangle \geq \|x\|\|u\|c\}$ the convex cone of those vectors u whose angle $\widehat{x\,u}$ with x, is at most θ.

Given $v, w \in K_c$, we have, by the triangle inequality, that $\widehat{v\,w} \leq \widehat{v\,x} + \widehat{x\,w} \leq 2\theta \leq \pi/2$. Thus

$$\cos\widehat{v\,w} \geq \cos\left(\widehat{v\,x} + \widehat{x\,w}\right) \geq \cos 2\theta = 1 - 2s^2 \geq 0.$$

And so, it is enough to show that $B_{c\|x\|}(x) \subseteq K_c$ or, equivalently, that $\operatorname{dist}(x, \partial K_c) \leq c\|x\|$.

Now, $\operatorname{dist}(x, \partial K_c) = \min\{\|x-u\| \mid \langle x, u\rangle = \|x\|\|u\|c\}$ where the latter equals the distance of x to a line having an angle θ with x, which is $\|x\|s$. □

5 Condition number

As other numerical algorithms in computational geometry, the Plantinga-Vegter algorithm has a cost which significantly varies with inputs of the same bit size. One wants to explain this variation in terms of geometric properties of the input. Condition numbers allow for such an explanation.

Definition 5.1 [5,13,18] Given $f \in \mathcal{H}_{n,d}$, the *local condition number* of f at $y \in \mathbb{S}^n$ is

$$\kappa(f, y) := \frac{\|f\|}{\sqrt{f(y)^2 + \frac{1}{d}\|\partial_y f_{|\mathrm{T}_y\mathbb{S}^n}\|^2}}.$$

Given $f \in \mathcal{P}_{n,d}$, the *local affine condition number* of f at $x \in \mathbb{R}^n$ is

$$\kappa_{\mathsf{aff}}(f, x) := \kappa(f^{\mathsf{h}}, \mathrm{IO}(x)).$$

5.1 What does κ_{aff} measure?

The nearer the hypersurface $V_{\mathbb{R}}(f)$ is to having a singularity at $x \in \mathbb{R}^n$, the smaller are the boxes drawn by the Plantinga-Vegter algorithm around x. Instead of controlling how near x is of being a singularity of f, we perform a Copernican turn and we control instead how near f is of having a singularity at x. This is precisely what $\kappa_{\mathsf{aff}}(f,x)$ does.

Theorem 5.1 (Condition Number Theorem) *Let $x \in \mathbb{R}^n$ and*

$$\Sigma_x := \{g \in \mathcal{P}_{n,d} \mid g(x) = 0,\, \partial_x g = 0\} \tag{5.1}$$

be the set of polynomials in $\mathcal{P}_{n,d}$ that have a singularity at x. Then for every $f \in \mathcal{P}_{n,d}$,

$$\frac{\|f\|}{\kappa_{\mathsf{aff}}(f,x)} = \operatorname{dist}(f, \Sigma_x)$$

where dist *is the distance induced by the Weyl norm on $\mathcal{P}_{n,d}$.*

Proof This is a reformulation of [5, Theorem 4.4] (cf. [4, Proposition 19.6]). □

Theorem 5.1 provides a geometric interpretation of the local condition number, and a corresponding "intrinsic" complexity parameter as desired by Burr, Gao and Tsigaridas in [10,11]. The next result is an essential tool for the probabilistic analyses. Note that, in the case under consideration, Σ_x is a linear subspace of codimension $n+1$ inside $\mathcal{P}_{n,d}$.

Corollary 5.1 *Let $x \in \mathbb{R}^n$ and let $\mathrm{R}_x : \mathcal{P}_{n,d} \to \Sigma_x^{\perp}$ be the orthogonal projection onto the orthogonal complement of the linear subspace Σ_x. Then*

$$\kappa_{\mathsf{aff}}(f,x) = \frac{\|f\|}{\|\mathrm{R}_x f\|}.$$

Proof We have that $\operatorname{dist}(f,\Sigma_x) = \|\mathrm{R}_x f\|$ since Σ_x is a linear subspace. Hence Theorem 5.1 finishes the proof. □

The equality in Corollary 5.1 should not come as a surprise, since κ_{aff} is defined in a way that the denominator is the norm of a vector depending linearly on f.

5.2 Regularity inequality

After doing our Copernican turn, we can control how near is $f \in \mathcal{P}_{n,d}$ of having a singularity at $x \in \mathbb{R}^n$. The regularity inequality [6, Proposition 3.6] (cf. [34, Proposition 1§2 3]) allows us to recover how near is x of being a singularity of f. More precisely, the regularity inequality gives lower bounds for the value of the function or its derivative in terms of the condition number.

Proposition 5.1 (Regularity inequality) *Let $f \in \mathcal{P}_{n,d}$ and $x \in \mathbb{R}^n$. Then either*

$$\left|\widehat{f}(x)\right| > \frac{1}{2\sqrt{2d}\,\kappa_{\mathsf{aff}}(f,x)} \quad \text{or} \quad \left\|\widehat{\partial f}(x)\right\| > \frac{1}{2\sqrt{2d}\,\kappa_{\mathsf{aff}}(f,x)}.$$

Proof Without loss of generality assume that $\|f\| = 1$. Let $y := \text{Ю}(x)$, $g := f^{\mathsf{h}}$ and assume that the first inequality does not hold. Then, by (4.2),

$$|g(y)| \leq \frac{1}{2\sqrt{2d}\,\kappa(g,y)\sqrt{1+\|x\|^2}}.$$

Now,

$$\frac{1}{\sqrt{2}\kappa(g,y)} \leq \max\left\{|g(y)|, \frac{1}{\sqrt{d}}\|\partial_y g_{\mathrm{T}_y\mathbb{S}^n}\|\right\} = \frac{1}{\sqrt{d}}\|\partial_y g_{\mathrm{T}_y\mathbb{S}^n}\|,$$

since $|g(y)| < \frac{1}{\sqrt{2}\kappa(g,y)}$. Thus, by (4.3) and (4.5), we get

$$\frac{1}{\sqrt{2}\,\kappa(F,y)} \leq \left\| \frac{\partial_x f}{(1+\|x\|^2)^{d/2}} - \frac{df(x)x^T}{(1+\|x\|^2)^{d/2+1}} \right\| \left(\frac{1+\|x\|^2}{\sqrt{d}} \right).$$

We divide by $\sqrt{d}$ and use the triangle inequality to obtain

$$\begin{aligned} \frac{1}{\sqrt{2d}\,\kappa(F,y)} &\leq \frac{\|\partial_x f\|}{d(1+\|x\|^2)^{d/2-1}} + \frac{|f(x)|}{(1+\|x\|^2)^{(d-1)/2}} \frac{\|x\|}{\sqrt{1+\|x\|^2}} \\ &= \left\| \widehat{\partial f}(x) \right\| + \left| \widehat{f}(x) \right| \frac{\|x\|}{\sqrt{1+\|x\|^2}}. \end{aligned}$$

By our assumtption and $\|x\| < \sqrt{1+\|x\|^2}$, the above inequality implies

$$\frac{1}{\sqrt{2d}\,\kappa(F,y)} < \left\| \widehat{\partial f}(x) \right\| + \frac{1}{2\sqrt{2d}\,\kappa_{\mathsf{aff}}(f,x)},$$

from where the desired inequality follows. □

6 Complexity Analysis of the Interval version

We analyze the complexity of PV-Interval in terms of the number of arithmetic operations the algorithm performs. This task reduces to estimating the number of boxes in the final subdivision produced by the algorithm. At the interval level, this is so, because each iteration of the algorithm takes the same number of arithmetic operations and the number of iterations is bounded by twice the number of final cubes. This was the underlying strategy in [10].

6.1 Local size bound framework

The original analysis in [10] was based on the notion of local size bound.

Definition 6.1 A *local size bound* for $C : \square\mathbb{R}^n \to \{\mathtt{True}, \mathtt{False}\}$ is a function $b : \mathbb{R}^n \to [0, \infty)$ such that for all $x \in \mathbb{R}^n$,

$$b(x) \leq \inf\{\mathrm{vol}(B) \mid x \in B \in \square\mathbb{R}^n \text{ and } C(B) \ \mathtt{False}\}.$$

The idea behind the local size bound is that it gives us the size from which every box containing x satisfies C. In our case, we will apply this to the condition C'_f introduced in Theorem 2.1.

Arguing as in [10, Proposition 4.1], one can easily get the following general bound. Although the proposition 6.1 is stated here for the particular case of the PV-algorithm, it's a general fact of subdivision based algorithms.

Proposition 6.1 *[10] The number of boxes in the final subdivision $\mathcal{S}$ returned by* PV-Interval *on input (f, a) is at most*

$$(2a)^n / \inf\{b(x) \mid x \in [-a, a]^n\} \tag{6.1}$$

where b is a local size bound for C'_f (of Theorem 2.1). □

The bound above is quite conservative as it considers the worst $b(x)$ over the $x \in [-a, a]^n$. Continuous amortization developed by Burr, Krahmer and Yap [8, 9], provides the following refined complexity estimate [10, Proposition 5.2] which is adaptive.

Theorem 6.1 *[8, 9, 10] The number of boxes in the final subdivision $\mathcal{S}$ returned by* PV-Interval *on input (f, a) is at most*

$$\max\left\{1, \int_{[-a,a]^n} \frac{2^n}{b(x)} \,\mathrm{d}x\right\}$$

where b is a local size bound for C'_f (of Theorem 2.1). Moreover, the bound is finite if and only if the algorithm terminates. □

To effectively use Theorem 6.1 we need explicit constructions for the local size bound.

6.2 Condition-based local size bound and complexity

The following result expresses a local size bound for C'_f in terms of the local condition number $\kappa_{\mathsf{aff}}(f,x)$.

Theorem 6.2 *The map*

$$x \mapsto 1/\left(2^{5/2}dn\kappa_{\mathsf{aff}}(f,x)\right)^n$$

is a local size bound for C'_f (of Theorem 2.1).

Proof Let $x \in \mathbb{R}^n$. Since $x \in B$, $\|x - m(B)\| \leq \sqrt{n}w(B)/2$. Hence, by Lemma 4.1 and the regularity inequality (Proposition 5.1), either

$$\left|\widehat{f}(m(B))\right| \geq \frac{1}{2\sqrt{2d}\,\kappa_{\mathsf{aff}}(f,x)} - (1+\sqrt{d})\sqrt{n}\,w(B)/2$$

or

$$\left|\widehat{\partial f}(m(B))\right| \geq \frac{1}{2\sqrt{2d}\,\kappa_{\mathsf{aff}}(f,x)} - (1+\sqrt{d-1})\sqrt{n}\,w(B)/2.$$

This means that $C'_f(B)$ is true if either

$$2\sqrt{2d}\,(1+\sqrt{d})\sqrt{n}\,\kappa_{\mathsf{aff}}(f,x)w(B) < 1$$

or

$$2\sqrt{2d}\,(1+\sqrt{d-1})n\kappa_{\mathsf{aff}}(f,x)w(B) < 1.$$

Hence we get that $C'_f(B)$ is true when both conditions are satisfied and the inequality $1+\sqrt{d} \leq 2\sqrt{d}$ finishes the proof. □

Using the results above, we get the following theorem exhibiting a condition-based complexity analysis of Algorithm 1.

Theorem 6.3 *The number of boxes in the final subdivision $\mathcal{S}$ of* PV-Interval *on input (f,a) is at most*

$$d^n \max\{1, a^n\} 2^{n\log n + \frac{9}{2}n}\, \mathbb{E}_{\mathfrak{x}\in[-a,a]^n}\left(\kappa_{\mathsf{aff}}(f,\mathfrak{x})^n\right).$$

The number of arithmetic operations performed by PV-Interval *on input (f,a) is at most*

$$\mathcal{O}\left(d^{n+1}\max\{1,a^n\}2^{n\log n+\frac{9}{2}n}N\,\mathbb{E}_{\mathfrak{x}\in[-a,a]^n}\left(\kappa_{\mathsf{aff}}(f,\mathfrak{x})^n\right)\right).$$

Proof It follows from Theorems 6.1 and 6.2 combined with the fact that $\int_{[-a,a]^n}\kappa_{\mathsf{aff}}(f,x)^n\,\mathrm{d}x$ equals $(2a)^n\,\mathbb{E}_{\mathfrak{x}\in[-a,a]^n}\left(\kappa_{\mathsf{aff}}(f,\mathfrak{x})^n\right)$. The latter follows from the fact that one performs $\mathcal{O}(dN)$ arithmetic operations to test C'_f and that the number of boxes that the algorithm generates is at most two times the number of final boxes. □

The above condition-based complexity estimate will become the main tool to prove Theorem 3.1 in Section 8 where we will study the quantity $\mathbb{E}_{\mathfrak{x}\in[-a,a]^n}\left(\kappa_{\mathsf{aff}}(f,\mathfrak{x})^n\right)$ for random $\mathfrak{f}$.

In the literature on numerical algorithms in real algebraic geometry [5,6,3,15,16,17,18], it is customary the use the following condition number

$$\kappa_{\mathsf{aff}}(f) := \max_{x\in[-a,a]^n}\kappa_{\mathsf{aff}}(f,x).$$

The quantity $\mathbb{E}_{\mathfrak{x}\in[-a,a]^n}\left(\kappa_{\mathsf{aff}}(f,\mathfrak{x})^n\right)$ in Theorem 6.3 is an average quantity, where else the customary condition number $\kappa_{\mathsf{aff}}(f)$ is a global suprema. The average quantity has finite expectation (over $\mathfrak{f}$), where the global suprema does not admit bounded first moment. This shows that a condition-based precision control combined with adaptive complexity techniques such as continuous amortization may lead to substantial improvements in computational real algebraic geometry.

6.3 Interlude: Complexity of the interval version of [10]

In [10], Burr, Gao and Tsigaridas gave an interval version of PV-Abstract different from PV-Interval based in the BGT interval approximation which relies on Taylor series. We provide a condition-based and probabilistic complexity analysis of this algorithm, although only for the interval version, on which we only bound the number of cubes and not the number of arithmetic operations.

We recall that Burr, Gao and Tsigaridas [10] showed that

$$\mathcal{C}(f,x) := \min\left\{\frac{2^{n-1}d/\ln\left(1+2^{2-2n}\right)+\sqrt{n}/2}{\operatorname{dist}(x,V_{\mathbb{C}}(f))}, \frac{2^{2n}(d-1)/\ln\left(1+2^{2-4n}\right)+\sqrt{n/2}}{\operatorname{dist}((x,x),V_{\mathbb{C}}(g_f))}\right\}$$

where g_f is the polynomial $\langle\partial f(X),\partial f(Y)\rangle$, is a local size bound for the condition that their interval version of PV-Abstract checks.

Theorem 6.4 *[10] The maps*

$$x \mapsto 1/\mathcal{C}(f,x)^n$$

is a local size bound function for the condition that the BGT interval version of PV-Abstract *checks.* □

Looking at the definition of $\mathcal{C}(f,x)$ in [10] one can see that $1/\mathcal{C}$ measures how near is x of being a singular zero of f. This is similar to $1/\kappa_{\mathsf{aff}}$ which, by Theorem 5.1, measures how near is f of having x as a singular zero. The following result relates these two quantities.

Theorem 6.5 *Let $d>1$ and $f\in\mathcal{P}_{n,d}$. Then, for all $x\in\mathbb{R}^n$,*

$$\mathcal{C}(f,x)\leq 2^{3n}d^2\kappa_{\mathsf{aff}}(f,x).$$

Proof Note that Lemma 4.1 holds over the complex numbers as well. Due to this and the fact that $V_{\mathbb{C}}(f)=V_{\mathbb{C}}(\widehat{f})$, we have that

$$\left|\widehat{f}(x)\right|\leq(1+\sqrt{d})\operatorname{dist}(x,V_{\mathbb{C}}(f)).$$

Now, if $\sqrt{2}(1+\sqrt{d-1})\operatorname{dist}((y_1,y_2),(x,x))<\|\widehat{\partial f}(x)\|$, then $\sqrt{2}(1+\sqrt{d-1})\|y_i-x\|<\|\widehat{\partial f}(x)\|$. Thus, by Lemma 4.1, $\sqrt{2}\left\|\widehat{\partial f}(y_i)-\widehat{\partial f}(x)\right\|<\left\|\widehat{\partial f}(x)\right\|$ and so, by Lemma 4.2, $0\neq\langle\widehat{\partial f}(y_1),\widehat{\partial f}(y_2)\rangle$. Hence

$$\left\|\widehat{\partial f}(x)\right\|\leq\sqrt{2}(1+\sqrt{d-1})\operatorname{dist}(x,V_{\mathbb{C}}(g_f)).$$

The bound now follows from Proposition 5.1, together with $2^{3(n-1)}d+\sqrt{n}\leq 2^{3n-2}d$ and

$$\min\left\{\frac{2^{n-1}d}{\ln\left(1+2^{2-2n}\right)}+\frac{\sqrt{n}}{2},\frac{2^{2n}(d-1)}{\ln\left(1+2^{2-4n}\right)}+\sqrt{\frac{n}{2}}\right\}\leq 2^{3n-4}d+\frac{\sqrt{n}}{2}.$$

The latter follows from

$$\frac{1}{\ln\left(1+2^{2-2n}\right)}\leq 2^{2n-3} \quad\text{and}\quad \frac{1}{\ln\left(1+2^{2-4n}\right)}\leq 2^{4n-3}, \tag{6.2}$$

which are deduced from first-order approximations of the natural logarithm. □

Theorems 6.4 and 6.5 combined with the continuous amortization of Burr, Krahmer and Yap [8, 9] gives the following complexity analysis.

Corollary 6.1 *The number of boxes in the final subdivision $\mathcal{S}$ of the BGT interval version of Algorithm 1 on input (f,a) is at most*

$$d^{2n}\max\{1,a^n\}2^{3n^2+2n}\,\mathbb{E}_{x\in[-a,a]^n}\left(\kappa_{\mathsf{aff}}(f,x)^n\right).$$

Remark 6.1 The main difference between $C(f,x)$ and $\kappa(f,x)$ is that $C(f,x)$ is a non-linear quantity and is hard to compute and to analyze, while the local condition number $\kappa(f,x)$—as indicated in Corollary 5.1—is a linear quantity, easier to compute and analyze.

We finish this interlude giving the probabilistic consequence of the above corollary. For proving it, one only has to use the techniques from Sections 8.

Theorem 6.6 *(A) Let $\mathfrak{f} \in \mathcal{P}_{n,d}$ be a dobro random polynomial with parameters K and ρ. The expected number of boxes in the final subdivision $\mathcal{S}$ of the BGT interval version of* PV-Abstract *on input $(\mathfrak{f}, a)$ is at most*

$$d^{n^2} N^{\frac{n+1}{2}} \max\{1, a^n\} 2^{3n^2+n\log n+7n+\frac{15}{2}} (K\rho)^{n+1}.$$

(S) Let $f \in \mathcal{P}_{n,d}$, $\sigma > 0$, and $\mathfrak{g} \in \mathcal{P}_{n,d}$ a dobro random polynomial with parameters $K \geq 1$ and ρ . Then the expected number of boxes of the final subdivision $\mathcal{F}$ of the BGT interval version of PV-Abstract *on input $(\mathfrak{q}_\sigma, a)$ where $\mathfrak{q}_\sigma = f + \sigma\|f\|\mathfrak{g}$ is at most*

$$d^{n^2} N^{\frac{n+1}{2}} \max\{1, a^n\} 2^{3n^2+n\log n+7n+\frac{15}{2}} (K\rho)^{n+1}.$$

Proof Apply Corollaries 8.1 and 8.3 to Corollary 6.1. □

7 Error and complexity analysis of the effective version

For an arithmetic expression ϕ and a point $x \in \mathbb{R}$, we will denote by $\mathtt{fl}(\phi(x)) \in \mathbb{F}$ the value obtained when evaluating ϕ at $r(x) \in \mathbb{F}$ using floating-point finite precision. In general, our objective is to show that for such expressions ϕ in our algorithm we have, for some other expression $\psi(x)$ and some $k \in \mathbb{N}$,

$$\mathtt{fl}(\phi(x)) = \phi(x) + \psi(x)\theta_k$$

where θ_k is any number $\delta \in \mathbb{R}$ satisfying

$$|\delta| \leq \frac{k\mathbf{u}}{1 - k\mathbf{u}}.$$

Note, this requires $k\mathbf{u} < 1$. This is the general strategy in [25, Chapter 3].

We proceed by introducing a new error symbol which will make our manipulations easier, then we recall some fundamental numerical algorithms for computing inner product and monomials and we apply them to the computed quantities during the execution of algorithm PV-Effective.

7.1 The arithmetic of error accumulation

To ease the notation of [25, Chapter 3], we slightly rewrite the notation used there. Instead of using the symbol θ_k for controlling errors, we will use $(\!(k)\!)$ and we will allow non-integer values inside. A fundamental way in which our treatment will differ is that we will reform the interpretation of the use of $(\!(k)\!)$ to make manipulations easier, emphasizing the use of the symbol instead of the error bounds directly.

Let ϕ be some arithmetic expression. Whenever we write an expression of the form

$$\mathtt{fl}(\phi(x)) = \tilde{\phi}\left(x, (\!(t_1)\!), \ldots, (\!(t_\ell)\!)\right) \tag{7.1}$$

for some arithmetic expression $\tilde{\phi}$ and for some real numbers $t_1, \ldots, t_\ell \geq 1$, we will mean that, as long as $\max\{t_1, \ldots, t_\ell\}\mathbf{u} < 1/2$, we have

$$\mathtt{fl}(\phi(x)) = \tilde{\phi}\left(x, \tau_1, \ldots, \tau_\ell\right)$$

for some

$$\tau_1 \in \left[-\frac{t_1\mathbf{u}}{1-t_1\mathbf{u}}, \frac{t_1\mathbf{u}}{1-t_1\mathbf{u}}\right], \ldots, \tau_\ell \in \left[-\frac{t_\ell\mathbf{u}}{1-t_\ell\mathbf{u}}, \frac{t_\ell\mathbf{u}}{1-t_\ell\mathbf{u}}\right].$$

We note that in this notation we are allowing more freedom as we don't require $t_1, \ldots, t_\ell$ to be integers. Furthermore, and this will make it computationally as useful as Landau notation, we introduce the following additional, asymmetric, notation.

For $\max\{t_1, \ldots, t_\ell, t_1', \ldots, t_{\ell'}'\}\mathbf{u} < 1/2$, we write

$$\tilde{\phi}\left(x, (\!(t_1)\!), \ldots, (\!(t_\ell)\!)\right) = \tilde{\phi}'\left(x, (\!(t_1')\!), \ldots, (\!(t_{\ell'}')\!)\right) \tag{7.2}$$

whenever for every

$$x, \tau_1 \in \left[-\frac{t_1\mathbf{u}}{1-t_1\mathbf{u}}, \frac{t_1\mathbf{u}}{1-t_1\mathbf{u}}\right], \ldots, \tau_\ell \in \left[-\frac{t_\ell\mathbf{u}}{1-t_\ell\mathbf{u}}, \frac{t_\ell\mathbf{u}}{1-t_\ell\mathbf{u}}\right],$$

there exist

$$\tau_1' \in \left[-\frac{t_1'\mathbf{u}}{1-t_1'\mathbf{u}}, \frac{t_1'\mathbf{u}}{1-t_1'\mathbf{u}}\right], \ldots, \tau_{\ell'}' \in \left[-\frac{t_{\ell'}'\mathbf{u}}{1-t_{\ell'}'\mathbf{u}}, \frac{t_{\ell'}'\mathbf{u}}{1-t_{\ell'}'\mathbf{u}}\right]$$

such that

$$\tilde{\phi}\left(x, \tau_1, \ldots, \tau_\ell\right) = \tilde{\phi}'\left(x, \tau_1', \ldots, \tau_{\ell'}'\right).$$

This is consistent with notation (7.1) in the sense that if both (7.1) and (7.2) hold then $\mathtt{fl}(\phi(x)) = \tilde{\phi}'\left(x, (\!(t_1)\!), \ldots, (\!(t_\ell)\!)\right)$. This will allow us to mechanically perform the finite precision analysis using the following rules.

Proposition 7.1 *For all $x, y \geq 1$, the following holds for the error symbol:*

- *(E1)* *If $x \leq y$, $(\!(x)\!) = (\!(y)\!)$.*
- *(E2)* $(\!(x)\!) + (\!(y)\!) + (\!(x)\!)\,(\!(y)\!) = (\!(x+y)\!)$.
  In particular, $(\!(x)\!) + (\!(y)\!) = (\!(x+y)\!)$ *and* $(1+(\!(x)\!))(1+(\!(y)\!)) = 1 + (\!(x+y)\!)$.
- *(E3)* $(1+(\!(x)\!))^{-1} = 1 + (\!(2x)\!)$.
- *(E4)* $\sqrt{1+(\!(x)\!)} = 1 + (\!(x)\!)$.
- *(E5)* *For all $t \in \mathbb{R}$,* $t\,(\!(x)\!) = |t|\,(\!(x)\!) = (\!(\max\{1,|t|\}x)\!)$.
- *(E6)* *For all $t, t' \in \mathbb{R}$,* $t\,(\!(x)\!) + t'\,(\!(y)\!) = (|t|+|t'|)\,(\!(\max\{x,y\})\!)$.
- *(E7)* *For all $t, t' \in (0,\infty)$, if $t < t'$, then* $t\,(\!(x)\!) = t'\,(\!(x)\!)$.
- *(E8)* $|1+(\!(x)\!)\,| = 1 + (\!(x)\!)$

Proof This follows from [25, Lemmas 3.1 and 3.3] □

The definition and properties of $(\!(\ )\!)$ follow the lines of classical error analysis, as e.g., in [25, Chapter 3]. The presentation may differ in minor details which we have chosen for our own convenience.

7.2 Basic finite precision algorithms

The following two propositions show the nice properties of the numerical computations that underlie the algorithm PV-Effective. Their statements refer to three aspects: 1) the number of arithmetic operations performed, 2) error estimates for a given input, and 3) error estimates for approximate inputs. From these bounds we can obtain bit-complexity estimates, as floating-point operations take $\mathcal{O}(|\log \mathbf{u}|^2)$-time (this being non-tight, one can obtain better bounds using fast multiplication algorithms).

Proposition 7.2 *There is a numerical algorithm which, with input $x, y \in \mathbb{R}^m$, computes $\langle x, y\rangle$. This algorithm satisfies the following:*

- *(i) It performs $\mathcal{O}(m)$ arithmetic operations.*
- *(ii) On input $x, y \in \mathbb{F}^m$, the computed value $\mathtt{fl}(\langle x, y\rangle)$ satisfies*

$$\mathtt{fl}(\langle x, y\rangle) = \langle x, y\rangle + \langle |x|, |y|\rangle\,(\!(\log m + 2)\!), \tag{7.3}$$

  where $|x| = (|x_1|, \ldots, |x_n|)$.
- *(iii) Assume $\tilde{x}, \tilde{y} \in \mathbb{F}^m$ and $x, y \in \mathbb{R}^m$ are such that, for all i,*

$$\tilde{x}_i = x_i + t_i\,(\!(\epsilon)\!) \text{ and } \tilde{y}_i = y_i + t_i'\,(\!(\epsilon')\!)$$

  for some $t, t' \in [0,\infty)^m$ and $\epsilon, \epsilon' \geq 1$. Then the computed value $\mathtt{fl}(\langle \tilde{x}, \tilde{y}\rangle)$ satisfies

$$\mathtt{fl}(\langle \tilde{x}, \tilde{y}\rangle) = \langle x, y\rangle + \max\{\langle |x|, |y|\rangle, \langle |t|, |y|\rangle, \langle |x|, |t'|\rangle, \langle |t|, |t'|\rangle\}\,\left(\!\left(\log m + \epsilon + \epsilon' + 2\right)\!\right).$$

Proposition 7.3 *There is a numerical algorithm which, with input $x \in \mathbb{R}^m$, computes $\|x\|$. This algorithm satisfies the following:*

(i) *It performs* $\mathcal{O}(m)$ *arithmetic operations.*
(ii) *On input* $x \in \mathbb{F}^m$*, the computed value* $\mathtt{fl}(\|x\|)$ *satisfies*

$$\mathtt{fl}(\|x\|) = \|x\|(1 + (\!(\log m + 3)\!)). \tag{7.4}$$

(iii) *Assume* $\tilde{x} \in \mathbb{F}^m$ *and* $x \in \mathbb{R}^m$ *are such that, for all* i*,*

$$\tilde{x}_i = x_i + t_i\,(\!(\epsilon)\!)$$

for some $t \in [0, \infty)^m$ *and* $\epsilon \geq 1$*. Then the computed value* $\mathtt{fl}(\|\tilde{x}\|\rangle)$ *satisfies*

$$\mathtt{fl}(\|\tilde{x}\|) = \|x\| + \max\{\|x\|, \|t\|\}\,(\!(\log m + \epsilon + 3)\!)\,.$$

Proposition 7.4 *There is a numerical algorithm which, with input* $x \in \mathbb{R}^n$ *and* $\alpha \in \mathbb{N}$*, computes* x^α*. This algorithm satisfies the following:*

(i) *It performs* $\mathcal{O}(\log|\alpha|)$ *arithmetic operations.*
(ii) *On input* $x \in \mathbb{F}^n$*, the computed value* $\mathtt{fl}(x^\alpha)$ *satisfies*

$$\mathtt{fl}(x^\alpha) = \begin{cases} x^\alpha(1 + (\!(|\alpha| - 1)\!)) & \textit{if } |\alpha| > 1 \\ x^\alpha, & \textit{otherwise.} \end{cases}$$

(iii) *Assume that* $\tilde{x} \in \mathbb{F}^n$ *and* $x \in \mathbb{R}^n$ *are such that, for all* i*,*

$$\tilde{x}_i = x_i(1 + (\!(\epsilon)\!))$$

for some $t \in [0, \infty)^m$ *and* $\epsilon \geq 1$*. Then the computed value* $\mathtt{fl}(\tilde{x}^\alpha)$ *satisfies*

$$\mathtt{fl}(\tilde{x}^\alpha) = \begin{cases} x^\alpha(1 + (\!(|\alpha|(1+\epsilon) - 1)\!)), & \textit{if } \alpha \neq 0 \\ 1, & \textit{otherwise.} \end{cases}$$

Proof (Proof of Proposition 7.2) The algorithm will first perform all the products $x_i y_i$ and them perform their sum by recursively dividing the sum into

$$\sum_{i \in I} x_i y_i + \sum_{i \in I^{\complement}} x_i y_i$$

where I and its complement, $I^{\complement}$ have size almost equal, differing in at most one.

(i) We initially perform m products and then $m-1$ additions. Note that the latter is independent of how we achieve the final sum, we sum as we do to minimize the error.

(ii) We will prove using induction the stronger claim that for the above algorithm

$$\mathtt{fl}(\langle x, y\rangle) = \langle x, y\rangle + \langle |x|, |y|\rangle\,(\!(\lceil \log m\rceil + 1)\!)$$

where $\lceil x\rceil$ is the minimum integer bigger or equal than x. Note that the claim is true for $m = 1$ and $m = 2$.

By the recursive nature of the algorithm, we have that

$$\begin{aligned}
&\mathtt{fl}\left(\sum_{i=1}^m x_i y_i\right) \\
&\quad = \mathtt{fl}\left(\sum_{i \in I} x_i y_i\right) \tilde{+}\, \mathtt{fl}\left(\sum_{i \in I^{\complement}} x_i y_i\right) \\
&\quad = \left(\sum_{i \in I} x_i y_i + \left(\sum_{i \in I} |x_i||y_i|\right)(\!(\lceil \log|I|\rceil + 1)\!)\right. \\
&\qquad \left. + \sum_{i \in I^{\complement}} x_i y_i + \left(\sum_{i \in I^{\complement}} |x_i||y_i|\right)(\!(\lceil \log(n - |I|)\rceil + 1)\!)\right)(1 + (\!(1)\!)) && \text{(Induction)} \\
&\quad = \left(\sum_{i=1}^n x_i y_i + \left(\sum_{i=1}^n |x_i||y_i|\right)(\!(\log\max\{|I|, n - |I|\} + 1)\!)\right)(1 + (\!(1)\!)) && (E6) \\
&\quad = (\langle x, y\rangle + \langle |x|, |y|\rangle\,(\!(\lceil \log\max\{|I|, n - |I|\}\rceil + 1)\!))\,(1 + (\!(1)\!))
\end{aligned}$$

Now, when $|I|$ and $n-|I|$ differ in at most one, we have that

$$\lceil \log \max\{|I|, n-|I|\}\rceil + 1 \leq \lceil \log n\rceil.$$

Thus

$$\begin{aligned}
&= (\langle x,y\rangle + \langle |x|,|y|\rangle \left(\!\left(\lceil \log n\rceil\right)\!\right)) (1+ \left(\!\left(1\right)\!\right)) \\
&= \langle x,y\rangle + \langle x,y\rangle \left(\!\left(1\right)\!\right) + \langle |x|,|y|\rangle (\left(\!\left(\lceil \log n\rceil\right)\!\right) + \left(\!\left(\lceil \log n\rceil\right)\!\right)\left(\!\left(1\right)\!\right)) \\
&= \langle x,y\rangle + \langle |x|,|y|\rangle (\left(\!\left(\lceil \log n\rceil\right)\!\right) + \left(\!\left(1\right)\!\right) + \left(\!\left(\lceil \log n\rceil\right)\!\right)\left(\!\left(1\right)\!\right)) && (E1) \text{ and } \langle x,y\rangle \leq \langle |x|,|y|\rangle \\
&= \langle x,y\rangle + \langle |x|,|y|\rangle \left(\!\left(\lceil \log n\rceil + 1\right)\!\right) && (E2).
\end{aligned}$$

(iii) Note that

$$\begin{aligned}
\langle \tilde{x},\tilde{y}\rangle &= \langle x,y\rangle + \langle (t_i \left(\!\left(\epsilon\right)\!\right)), y\rangle + \langle x, (t_i' \left(\!\left(\epsilon'\right)\!\right))\rangle + \langle (t_i \left(\!\left(\epsilon\right)\!\right)), (t_i' \left(\!\left(\epsilon'\right)\!\right))\rangle \\
&= \langle x,y\rangle + \langle |t|,|y|\rangle \left(\!\left(\epsilon\right)\!\right) + \langle |x|,|t'|\rangle \left(\!\left(\epsilon'\right)\!\right) + \langle |t|,|t'|\rangle \left(\!\left(\epsilon\right)\!\right)\left(\!\left(\epsilon'\right)\!\right) && (E6) \\
&= \langle x,y\rangle + \max\{\langle |t|,|y|\rangle, \langle |x|,|t'|\rangle, \langle |t|,|t'|\rangle\}(\left(\!\left(\epsilon\right)\!\right) + \left(\!\left(\epsilon'\right)\!\right) + \left(\!\left(\epsilon\right)\!\right)\left(\!\left(\epsilon'\right)\!\right)) && (E7) \\
&= \langle x,y\rangle + \max\{\langle |t|,|y|\rangle, \langle |x|,|t'|\rangle, \langle |t|,|t'|\rangle\} \left(\!\left(\epsilon+\epsilon'\right)\!\right) && (E2)
\end{aligned}$$

An analogous statement holds for $\langle |\tilde{x}|, |\tilde{y}|\rangle$. Now, combining this and (ii), we get that

$$\begin{aligned}
&\mathtt{fl}(\langle \tilde{x},\tilde{y}\rangle) \\
&\quad= \langle \tilde{x},\tilde{y}\rangle + \langle |\tilde{x}|,|\tilde{y}|\rangle \left(\!\left(\log m + 2\right)\!\right) \\
&\quad= \langle x,y\rangle + \max\{\langle |t|,|y|\rangle, \langle |x|,|t'|\rangle, \langle |t|,|t'|\rangle\} \left(\!\left(\epsilon+\epsilon'\right)\!\right) \\
&\qquad + \left(\langle |x|,|y|\rangle + \max\{\langle |t|,|y|\rangle, \langle |x|,|t'|\rangle, \langle |t|,|t'|\rangle\} \left(\!\left(\epsilon+\epsilon'\right)\!\right)\right) \left(\!\left(\log m + 2\right)\!\right) \\
&\quad= \langle x,y\rangle \\
&\qquad\quad + \max\{\langle |x|,|y|\rangle, \langle |t|,|y|\rangle, \langle |x|,|t'|\rangle, \langle |t|,|t'|\rangle\} \\
&\qquad\qquad \cdot \left(\left(\!\left(\epsilon+\epsilon'\right)\!\right) + \left(\!\left(\log m + 2\right)\!\right) + \left(\!\left(\epsilon+\epsilon'\right)\!\right)\left(\!\left(\log m + 2\right)\!\right)\right) && (E7) \\
&\quad= \langle x,y\rangle + \max\{\langle |x|,|y|\rangle, \langle |t|,|y|\rangle, \langle |x|,|t'|\rangle, \langle |t|,|t'|\rangle\} \left(\!\left(\log m + \epsilon + \epsilon' + 2\right)\!\right).
\end{aligned}$$

□

Proof (Proof of Proposition 7.3) The proof is analogous to that of Proposition 7.2. □

Proof (Proof of Proposition 7.4) The proof is analogous to that of Proposition 7.2, but we have to take into account that errors accumulate additively since in each multiplication the errors of the computed quantities are added by (E2). □

7.3 Finite-precision computations

We study the errors due to finite-precision in algorithm PV-Effective and show its correctness. The following lemma is useful.

Lemma 7.1 *There is a numerical algorithm which, with input $f \in \mathcal{P}_{n,d}$, computes the Weyl norm $\|f\|$ of f. This algorithm performs $\mathcal{O}(N)$ arithmetic operations, and, on input $f \in \mathcal{P}_{n,d} \cap \mathbb{F}[X_1,\dots,X_n]$, the computed value $\mathtt{fl}(\|f\|)$ satisfies*

$$\mathtt{fl}(\|f\|) = \|f\|(1 + \left(\!\left(\log N + 8\right)\!\right)).$$

Moreover, for general $f \in \mathcal{P}_{n,d}$,

$$\mathtt{fl}(\|r(f)\|) = \|f\|(1 + \left(\!\left(\log N + 9\right)\!\right)).$$

Proof To compute the Weyl norm, we first compute the vector $\left(\binom{d}{\alpha}^{-1/2} f_\alpha\right)$ and then its norm. To compute the vector, we take the floating point approximation of $\binom{d}{\alpha}$, we compute its square root and we divide f_α by the computed square root. Hence

$$\begin{aligned}\mathtt{fl}\left(\binom{d}{\alpha}^{-1/2} f_\alpha\right) &= \binom{d}{\alpha}^{-1/2} f_\alpha \frac{(1+(\!(1)\!))}{\sqrt{1+(\!(1)\!)}(1+(\!(1)\!))} \\ &= \binom{d}{\alpha}^{-1/2} f_\alpha(1+(\!(5)\!)) \qquad \text{(Proposition 7.1)}\end{aligned}$$

Now, the lemma follows from Proposition 7.3. □

The following two propositions show that we can compute $|\widehat{f}|$ and $\left\|\widehat{\partial f}\right\|$ in a stable way.

Proposition 7.5 *There is a numerical algorithm which, with input $f \in \mathcal{P}_{n,d}$ and $x \in \mathbb{R}^n$, computes $|\widehat{f}(x)|$. This algorithm performs $\mathcal{O}(dN)$ arithmetic operations, and, on input $x \in \mathbb{F}^n$ and $f \in \mathcal{P}_{n,d} \cap \mathbb{F}[X_1, \dots, X_n]$, the computed value $\mathtt{fl}(|\widehat{f}(x)|)$ satisfies*

$$\mathtt{fl}(|\widehat{f}(x)|) = |\widehat{f}(x)| + \sqrt{1+\|x\|}\,(\!(32d\log(n+1))\!)\,.$$

In particular, if the round-off unit satisfies

$$\mathbf{u} \leq \frac{1}{64d\log(n+1)},$$

then for $x \in [-a,a]^n \cap \mathbb{F}^n$,

$$\left|\mathtt{fl}(|\widehat{f}(x)|) - |\widehat{f}(x)|\right| \leq 64\sqrt{2}d\sqrt{n+1}\log(n+1)\max\{1,a\}\mathbf{u}.$$

The above remains true for arbitrary f and x if we apply the algorithm to $r(f)$ and $r(x)$.

Proof We first compute $f(x)$ as $\langle (f_\alpha), (x^\alpha)\rangle$, where the x^α are computed one by one, and then divide the result by the computed $\|f\|\|(1,x)\|^{d-1}$ to obtain $\widehat{f}(x)$.

By Propositions 7.2 and 7.4 and (E7), we have that

$$\mathtt{fl}(f(x)) = f(x) + \|f\|\|(1,x)\|^d\,(\!(\log N + d + 1)\!)\,,$$

since $\langle (|f_\alpha|), (|x^\alpha|)\rangle = g(|x|)$, where $g = \sum_\alpha |f_\alpha| X^\alpha$, is bounded by $\|f\|\|(1,x)\|^d$, by Lemma 4.1.

Also, by Proposition 7.3, Lemma 7.1 and (E2), we have that

$$\mathtt{fl}\|f\|\|(1,x)\|^{d-1} = \|f\|\|(1,x)\|^{d-1}(1+(\!(\log N + d\log(n+1) + 4d + 2)\!)).$$

Now, $N \leq (n+1)^d$. Thus we have that

$$\mathtt{fl}(f(x)) = f(x) + \|f\|\|(1,x)\|^d\,(\!(3d\log(n+1))\!)$$

and

$$\mathtt{fl}(\|f\|\|(1,x)\|^{d-1}) = \|f\|\|(1,x)\|^{d-1}(1+(\!(8d\log(n+1))\!)).$$

Although, doing this we are not obtaining tight bounds, we have to recall that the number of digits is proportional to the logarithm of what is inside $(\!(\cdot)\!)$.

To finish, we only have to do the division. Thus

$$\begin{aligned}
&\mathtt{fl}(\widehat{f}(x))\\
&= \mathtt{fl}(f(x))/\mathtt{fl}(\|f\|\|(1,x)\|^{d-1})(1+(\!(1)\!))\\
&= (f(x)+\|f\|\|(1,x)\|^{d}\,(\!(3d\log(n+1))\!))/\left(\|f\|\|(1,x)\|^{d-1}(1+(\!(7d\log(n+1))\!))\right)(1+(\!(1)\!))\\
&= (\widehat{f}(x)+\|(1,x)\|\,(\!(3d\log(n+1))\!)\,(1+(\!(8d\log(n+1))\!))^{-1}(1+(\!(1)\!))\\
&= (\widehat{f}(x)+\|(1,x)\|\,(\!(3d\log(n+1))\!)\,(1+(\!(16d\log(n+1)+1)\!))\\
&= \widehat{f}(x)+\widehat{f}(x)\,(\!(10d\log(n+1)+1)\!)\\
&\quad+\|(1,x)\|(\,(\!(3d\log(n+1))\!)+(\!(3d\log(n+1))\!)\,(\!(14d\log(n+1)+1)\!))\\
&= \widehat{f}(x)\\
&\quad+\|(1,x)\|(\,(\!(16d\log(n+1)+1)\!)+(\!(3d\log(n+1))\!)+(\!(3d\log(n+1))\!)\,(\!(16d\log(n+1)+1)\!))\\
&= \widehat{f}(x)+\|(1,x)\|\,(\!(19d\log(n+1)+1)\!)\\
&= \widehat{f}(x)+\|(1,x)\|\,(\!(20d\log(n+1))\!)
\end{aligned}$$

where the first equality follows from how we compute $\widehat{f}(x)$, the second one from the above identities, the fourth one from (E3) and (E2), the sixth one from Lemma 4.1 and (E7), the eighth one from (E2), and the last one from (E1).

The result for $r(f)$ and $r(x)$ follows similarly. □

Proposition 7.6 *There is a round-off algorithm which, with input $f\in\mathcal{P}_{n,d}$ and $x\in\mathbb{R}^n$, computes $\|\widehat{\partial f}(x)\|$. It performs $\mathcal{O}(dN)$ arithmetic operations, and, on input $x\in\mathbb{F}^n$ and $f\in\mathcal{P}_{n,d}\cap\mathbb{F}[X_1,\dots,X_n]$, the computed value $\|\widehat{\partial f}(x)\|$ satisfies*

$$\mathtt{fl}(\|\widehat{\partial f}(x)\|)=\|\widehat{\partial f}(x)\|+\sqrt{1+\|x\|}\,(\!(32d\log(n+1))\!)\,.$$

In particular, if the round-off unit satisfies

$$\mathbf{u}\leq\frac{1}{64d\log(n+1)},$$

then for $x\in[-a,a]^n\cap\mathbb{F}^n$,

$$\left|\mathtt{fl}(\|\widehat{\partial f}(x)\|)-\|\widehat{\partial f}(x)\|\right|\leq 64\sqrt{2}d\sqrt{n+1}\log(n+1)\max\{1,a\}\mathbf{u}.$$

The above remains true for arbitrary f and x if we apply the algorithm to $r(f)$ and $r(x)$.

Proof We compute each $\partial_j f(x)$ as we computed $f(x)$. After that, we compute $\|\partial f(x)\|$, $d\|f\|\|(1,x)\|^{d-2}$ and their quotient.

By Propositions 7.2 and 7.4 and (E7), we have that

$$\mathtt{fl}(\partial_j f(x))=\partial_j f(x)+\partial_j g(|x|)\,(\!(\log N+d+1)\!)\,,$$

where $g=\sum_\alpha|f_\alpha|X^\alpha$. Now, by Proposition 7.3, we have that

$$\mathtt{fl}(\|\partial f(x)\|)=\|\partial f(x)\|+\max\{\|\partial f(x)\|,\partial g(|x|)\|\}\,(\!(\log N+\log n+d+4)\!)\,.$$

However, by Lemma 4.1, both $\|\partial f(x)\|$ and $\|\partial g(|x|)\|$ are bounded by $d\|f\|\|(1,x)\|^{d-1}$. Thus, by (E7),

$$\mathtt{fl}(\|\partial f(x)\|)=\|\partial f(x)\|+d\|f\|\|(1,x)\|^{d-1}\,(\!(\log N+\log n+d+4)\!)\,.$$

Again, by Proposition 7.3, Lemma 7.1 and (E2), we have that

$$\mathtt{fl}\left(d\|f\|\|(1,x)\|^{d-2}\right)=d\|f\|\|(1,x)\|^{d-2}(1+(\!(\log N+d\log(n+1)+4d+2)\!)).$$

Now, as $N\leq(n+1)^d$, we have

$$\mathtt{fl}(\|\partial f(x)\|)=\|\partial f(x)\|+d\|f\|\|(1,x)\|^{d-1}\,(\!(7d\log(n+1))\!)$$

and

$$\mathtt{fl}\left(d\|f\|\|(1,x)\|^{d-2}\right)=d\|f\|\|(1,x)\|^{d-2}(1+(\!(8d\log(n+1))\!)).$$

Now, arguing as in Proposition 7.5, the desired statement follows. □

We can now show the correctness of Algorithm PV-Effective. We will denote by $\mathtt{fl}(B)$ the rounding $r(B)$ of a box given by

$$m(\mathtt{fl}(B)) = m(B)(1 + ((1))) \quad \text{and} \quad w(\mathtt{fl}(B)) = w(B)(1 + ((1))).$$

Similarly, we will write $\mathtt{fl}(f)$ to denote the rounding $r(f)$ of f. The next theorem shows that if the round-off unit is sufficiently small, then a floating-point version of condition $C'_f(B)$ is good enough to check $C_f(B)$.

Theorem 7.1 *Let $B \in \square[-a,a]^n$. If*

$$C_f^{\mathrm{FP}} : \begin{cases} \mathtt{fl}\left(\left|\widehat{\mathtt{fl}(f)}(m(\mathtt{fl}(B)))\right|\right) > \mathtt{fl}\left(4\sqrt{d}\sqrt{n+1}w(\mathtt{fl}(B))\right) \\ \qquad\qquad \textit{or} \ \left(\left\|\widehat{\partial\mathtt{fl}(f)}(m(\mathtt{fl}(B)))\right\|\right) > \mathtt{fl}\left(6\sqrt{d}(n+1)w(\mathtt{fl}(B))\right) \end{cases}$$

and

$$\mathbf{u} \le \frac{1}{128\sqrt{dn}} \frac{\min\{1, w(B)\}}{\max\{1, a\}},$$

then $C'_f(B)$ holds and, hence, so does $C_f(B)$.

Corollary 7.1 *Algorithm* PV-Effective *is correct.*

Proof Note that the conditions of Propositions 7.5 and 7.6 are satisfied. Therefore we have that

$$|\widehat{f}(m(B))| > \mathtt{fl}\left(\left|\widehat{\mathtt{fl}(f)}(m(\mathtt{fl}(B)))\right|\right) - \sqrt{d}\log(n+1)\min\{1, w(B)\}$$

and that

$$\left\|\widehat{\partial f}(m(B))\right\| > \mathtt{fl}\left(\left\|\widehat{\partial\mathtt{fl}(f)}(m(\mathtt{fl}(B)))\right\|\right) - \sqrt{d}\log(n+1)\min\{1, w(B)\}$$

after taking into account the bound for $\mathbf{u}$.

By error analysis (Proposition 7.1), we have that

$$\mathtt{fl}\left(4\sqrt{d}\sqrt{n+1}w(\mathtt{fl}(B))\right) = 4\sqrt{d}\sqrt{n+1}w(B)(1 + ((8)))$$

and

$$\mathtt{fl}\left(4\sqrt{d}(n+1)w(\mathtt{fl}(B))\right) = 6\sqrt{d}(n+1)w(B)(1 + ((8))).$$

Hence

$$\mathtt{fl}\left(4\sqrt{d}\sqrt{n+1}w(\mathtt{fl}(B))\right) > 4\sqrt{d}\sqrt{n+1}w(B)\left(1 - \frac{1}{8\sqrt{dn}}\frac{\min\{1, w(B)\}}{\max\{1, a\}}\right)$$

and

$$\mathtt{fl}\left(4\sqrt{d}(n+1)w(\mathtt{fl}(B))\right) > 6\sqrt{d}(n+1)w(B)\left(1 - \frac{1}{8\sqrt{dn}}\frac{\min\{1, w(B)\}}{\max\{1, a\}}\right).$$

Combining the two pairs of inequalities above we get

$$\begin{aligned} |\widehat{f}(m(B))| &> 2\sqrt{d}\sqrt{n+1}w(B) \\ &\quad + 2\sqrt{d}\sqrt{n+1}w(B)\left(1 - \frac{1}{4\sqrt{dn}}\frac{\min\{1, w(B)\}}{\max\{1, a\}} - \frac{\log(n+1)}{2\sqrt{n+1}}\min\left\{1, \frac{1}{w(B)}\right\}\right) \end{aligned} \tag{7.5}$$

and

$$\begin{aligned} \left\|\widehat{\partial f}(m(B))\right\| &> 3\sqrt{d}(n+1)w(B) \\ &\quad + 3\sqrt{d}(n+1)w(B)\left(1 - \frac{1}{6\sqrt{dn}}\frac{\min\{1, w(B)\}}{\max\{1, a\}} - \frac{\log(n+1)}{2(n+1)}\min\left\{1, \frac{1}{w(B)}\right\}\right) \end{aligned} \tag{7.6}$$

Now, the term between parentheses in the right-hand side of (7.5) is positive since

$$\frac{1}{4\sqrt{dn}}\frac{\min\{1, w(B)\}}{\max\{1, a\}} + \frac{\log(n+1)}{2\sqrt{n+1}}\min\left\{1, \frac{1}{w(B)}\right\} \le \frac{1}{4\sqrt{dn}} + \frac{\log(n+1)}{2\sqrt{n+1}} \le \frac{1}{4} + \frac{1}{2} < 1,$$

and so is the one in the right-hand side of (7.6) since

$$\frac{1}{6\sqrt{dn}}\frac{\min\{1, w(B)\}}{\max\{1, a\}} + \frac{\log(n+1)}{2(n+1)}\min\left\{1, \frac{1}{w(B)}\right\} \le \frac{1}{6\sqrt{dn}} + \frac{1}{2\sqrt{n+1}} \le \frac{1}{6} + \frac{1}{2\sqrt{2}} < 1.$$

Therefore our claim holds. □

7.4 Complexity of Algorithm PV-Effective

We now prove the analogous of Theorem 6.2 in the finite-precision setting. To do so we have to slightly modify the sense of the term 'local size bound' to take finite precision into account.

Definition 7.1 A *local size bound* for C_f^{FP} is a function $b_f^{\mathrm{FP}}:\mathbb{R}^n\to[0,\infty)$ such that for all $x\in\mathbb{R}^n$,

$$b_f^{\mathrm{FP}}(x)\leq\inf\left\{\mathrm{vol}(B)\mid x\in B\in\square\mathbb{R}^n,\, C_f^{\mathrm{FP}}(B)\ \texttt{False}\text{ with }\mathbf{u}\leq\frac{1}{128\sqrt{dn}}\frac{\min\{1,w(B)\}}{\max\{1,a\}}\right\}.$$

The modifications takes into account that the condition C_f^{FP} is checked with sufficiently large precision, as indicated by Theorem 7.1. The theorem below gives us the local size bound for finite precision.

Theorem 7.2 *The map*

$$x\mapsto 1/\left(2^6dn\kappa_{\mathsf{aff}}(f,x)\right)^n$$

is a local size bound for C_f^{FP} (of Theorem 7.1).

Proof The proof is similar to the one of Theorem 7.1. For now on, let $B\in\square\mathbb{R}^n$ be such that $x\in B$.

By Proposition 7.5 and 7.6 and the bound on $\mathbf{u}$, we have that

$$\texttt{fl}\left(\left|\widehat{\texttt{fl}(f)}(m(\texttt{fl}(B)))\right|\right)>\left|\widehat{f}(m(B))\right|-\sqrt{d}\log(n+1)\min\{1,w(B)\}$$

and that

$$\texttt{fl}\left(\left\|\widehat{\partial\texttt{fl}(f)}(m(\texttt{fl}(B)))\right\|\right)>\left\|\widehat{\partial f}(m(B))\right\|-\sqrt{d}\log(n+1)\min\{1,w(B)\}.$$

By error analysis (Proposition 7.1),

$$4\sqrt{d}\sqrt{n+1}w(B)\left(1+\frac{1}{8\sqrt{dn}}\frac{\min\{1,w(B)\}}{\max\{1,a\}}\right)>\texttt{fl}\left(4\sqrt{d}\sqrt{n+1}w(\texttt{fl}(B))\right)$$

and

$$6\sqrt{d}(n+1)w(B)\left(1+\frac{1}{8\sqrt{dn}}\frac{\min\{1,w(B)\}}{\max\{1,a\}}\right)>\texttt{fl}\left(4\sqrt{d}(n+1)w(\texttt{fl}(B))\right).$$

By the regularity inequality (Proposition 5.1) and Corollary 4.1, we know that either

$$\begin{aligned}&\texttt{fl}\left(\left|\widehat{\texttt{fl}(f)}(m(\texttt{fl}(B)))\right|\right)\\&\quad>\frac{1}{2\sqrt{2}d\kappa_{\mathsf{aff}}(f,x)}-\frac{(1+\sqrt{d})\sqrt{n}}{2}w(B)-\sqrt{d}\log(n+1)\min\{1,w(B)\}\\&\quad>\frac{1}{2\sqrt{2}d\kappa_{\mathsf{aff}}(f,x)}-2\sqrt{dn}w(B)\end{aligned}$$

or

$$\begin{aligned}&\texttt{fl}\left(\left\|\widehat{\partial\texttt{fl}(f)}(m(\texttt{fl}(B)))\right\|\right)\\&\quad>\frac{1}{2\sqrt{2}d\kappa_{\mathsf{aff}}(f,x)}-\frac{(1+\sqrt{d-1})\sqrt{n}}{2}w(B)-\sqrt{d}\log(n+1)\min\{1,w(B)\}\\&\quad>\frac{1}{2\sqrt{2}d\kappa_{\mathsf{aff}}(f,x)}-2\sqrt{dn}w(B).\end{aligned}$$

Hence $C_f^{\mathrm{FP}}(B)$ holds as long as

$$\frac{1}{2\sqrt{2}d\kappa_{\mathsf{aff}}(f,x)}-2\sqrt{dn}w(B)>6\sqrt{d}(n+1)w(B)\left(1+\frac{1}{8\sqrt{dn}}\frac{\min\{1,w(B)\}}{\max\{1,a\}}\right),$$

which is implied by

$$2^6d(n+1)\kappa_{\mathsf{aff}}(f,x)w(B)<1.$$

This means that $C_f^{\mathrm{FP}}(B)$ is true when $\mathrm{vol}(B)<1/\left(2^6dn\kappa_{\mathsf{aff}}(f,x)\right)^n$, which is what we wanted to show. □

Using the continuous amarotization [8,9] (we use the statement in [11, Theorem 5]), we obtain the following condition-based complexity analysis of PV-EFFECTIVE.

Theorem 7.3 *The number of boxes in the final subdivision $\mathcal{S}$ of* PV-EFFECTIVE *on input (f,a) is at most*

$$d^n a^n 2^{n\log n+8n}\,\mathbb{E}_{\mathfrak{x}\in[-a,a]^n}\left(\kappa_{\mathsf{aff}}(f,\mathfrak{x})^n\right).$$

The number of arithmetic operations performed by PV-EFFECTIVE *on input (f,a) is at most*

$$\mathcal{O}\left(d^{n+1}a^n 2^{n\log n+8n}N\,\mathbb{E}_{\mathfrak{x}\in[-a,a]^n}\left(\kappa_{\mathsf{aff}}(f,\mathfrak{x})^n\right)\right).$$

Furthermore, the bit-cost of PV-EFFECTIVE *on input (f,a) is at most*

$$\mathcal{O}\left(d^{n+1}a^n 2^{n\log n+8n}N\log^2(dna)\,\mathbb{E}_{\mathfrak{x}\in[-a,a]^n}\left(\kappa_{\mathsf{aff}}(f,\mathfrak{x})^n\log^2\kappa_{\mathsf{aff}}(f,x)\right)\right)$$

under the assumptions that floating-point arithmetic is done using standard arithmetic and that that the cost of operating with the exponents is negligible.

Proof The first two points follow from Theorems 7.2 and 6.1. For the third point, we recall the following variant of Theorem 6.1 that can be found in [11, Theorem 5]. Let $\mathcal{S}$ be the final subdivision output by PV-INTERVAL and $h:(0,\infty)\to(0,\infty)$ a continuous map. Then

$$\sum_{B\in\mathcal{S}} h\left(w(B)\right)\le\max\left\{h(2a),\int_{[-a,a]^n}\frac{2^n}{b_f^{\mathrm{FP}}(x)}\,h\left(\frac{b_f^{\mathrm{FP}}(x)^{\frac{1}{n}}}{2}\right)\mathrm{d}x\right\}.$$

Applying Theorem 7.2, we get the bound

$$\sum_{B\in\mathcal{S}} h\left(w(B)\right)\le\max\left\{h(2a),2^{n\log n+7n}d^n\int_{[-a,a]^n}\kappa_{\mathsf{aff}}(f,x)^n h\left(2^5dn\kappa_{\mathsf{aff}}(f,x)\right)\mathrm{d}x\right\}.$$

Now, we note that testing C_f^{FP} at each of the boxes along the way takes at most $\mathcal{O}(dN)$ arithmetic operations and that the number of boxes that the algorithm deals with is at most twice the number of final boxes. Because of this, the bit-cost of the algorithm (ignoring the cost of operating with exponents) in floating-point arithmetic is

$$\mathcal{O}\left(dN\sum_{B\in\mathcal{S}}\mathbf{m}_B^2\right).$$

This is so, because each arithmetic operation takes $\mathcal{O}(\mathbf{m}^2)$ bit-time and $\mathbf{m}_B$ is the maximum precision needed to test C_f^{FP} in any box that is an ancestor of B. Hence, taking

$$h(w(B))=\mathcal{O}\left(\max\left\{\log^2 2^9\sqrt{dn}a,\log^2 2^9\sqrt{dn}\frac{a}{w(B)}\right\}\right)$$

gives the final bound. □

The above condition-based complexity estimate will become the complexity estimates in Theorem 3.2 in the coming Section 8.

8 Probabilistic analyses

In this section, we prove Theorems 3.1 and 3.2 stated in Section 3 using Theorems 6.2 and 7.2 respectively.

8.1 Probabilistic toolbox

We now introduce our probabilistic toolbox. The main tools are a tail bound on the norm of a random vector and a small ball type estimate to ensure norm of a random projection is not too small. Following [34, $5^{\S 1}$], we will give explicit constants avoiding the use of undefined absolute constants. This will require us to sketch some proofs.

Theorem 8.1 *Let $\mathfrak{x} \in \mathbb{R}^N$ be a random vector where each component $\mathfrak{x}_i$ is centered and sub-Gaussian with ψ_2-norm K. Then for all $t \geq 5K\sqrt{N}$,*

$$\mathbb{P}\left(\|\mathfrak{x}\| \geq t\right) \leq \exp\left(-\frac{t^2}{(5K)^2}\right). \tag{8.1}$$

Proof (Sketch of proof) We follow the ideas in [35, Theorems 2.6.3]. Note that $\|\mathfrak{x}\| \geq t$ is equivalent to $e^{s^2\|\mathfrak{x}\|^2} \geq e^{s^2t^2}$. By Markov's inequality and independence,

$$\mathbb{P}\left(\|\mathfrak{x}\| \geq t\right) \leq e^{-s^2t^2}\mathbb{E}e^{s^2\|\mathfrak{x}\|^2} = \prod_{i=1}^{N}\mathbb{E}e^{s^2\mathfrak{x}_i^2}.$$

By assumption, for each i,

$$\mathbb{E}e^{s^{2l}\mathfrak{x}_i^2} = \sum_{l=0}^{\infty}\frac{s^{2l}\mathbb{E}\mathfrak{x}_i^{2l}}{l!} \leq \sum_{l=1}^{\infty}\frac{s^{2l}K^{2l}(2l)^l}{l!} \leq \sum_{l=0}^{\infty}\left(2eK^2s^l\right)^l,$$

since $l! \geq (l/e)^l$. Thus, taking $s^2 = 1/(4eK^2)$, we get

$$\mathbb{P}\left(\|\mathfrak{x}\| \geq t\right) = 2^N e^{-t^2/(4eK^2)}.$$

The claim is now trivial assuming $t \geq \sqrt{8e\ln(2)}K\sqrt{N}$. □

Theorem 8.2 *[31, Corollary 1.4] Let $\mathfrak{x} \in \mathbb{R}^N$ be a random vector where each component $\mathfrak{x}_i$ has the anti-concentration property with constant ρ and $P : \mathbb{R}^N \to \mathbb{R}^N$ an orthogonal projection onto a k-dimensional linear subspace of $\mathbb{R}^N$. Then for all $\varepsilon > 0$,*

$$\mathbb{P}\left(\|P\mathfrak{x}\| \leq \sqrt{k}\varepsilon\right) \leq (3\rho\varepsilon)^k.$$

Proof (Sketch of proof) Note that by assumption, each $\mathfrak{x}_i$ has probability density (with respect to the Lebesgue measure) bounded by $\rho/2$. Then, by [26, Theorem 1.1.], $P\mathfrak{x}$ has probability density (with respect to the Lebesgue measure) bounded by $\left(\rho/\sqrt{2}\right)^k$. Thus

$$\mathbb{P}\left(\|P\mathfrak{x}\| \leq \sqrt{k}\varepsilon\right) \leq \omega_k\left(\frac{\sqrt{k}\rho}{\sqrt{2}}\right)^k$$

where ω_k is the volume of the k-dimensional Euclidean ball.

Now, $\omega_k k^{\frac{k}{2}} \leq (2e)^{\frac{k}{2}}\pi^{\frac{k}{2}}$, from where the claim follows. □

8.2 Average Complexity Analysis

The following theorem is the main technical result from which the average complexity bound will follow.

Theorem 8.3 *Let $\mathfrak{f} \in \mathcal{P}_{n,d}$ be a dobro random polynomial with parameters K and ρ. For all $x \in \mathbb{R}^n$ and $t \geq e$,*

$$\mathbb{P}\left(\kappa_{\mathsf{aff}}(\mathfrak{f}, x) \geq t\right) \leq 2\left(\frac{N}{n+1}\right)^{\frac{n+1}{2}}(15K\rho)^{n+1}\frac{\ln(t)^{\frac{n+1}{2}}}{t^{n+1}}.$$

Remark 8.1 By [21, (1)], we have $K\rho \geq \frac{1}{4}$ for a dobro random polynomial $\mathfrak{f}$ with parameters K and ρ. This fact will be used without mention in the bounds below.

Proof (Proof of Theorem 8.3) By Corollary 5.1, we have that $\kappa_{\mathsf{aff}}(\mathfrak{f},x) = \|\mathfrak{f}\|/\|\mathrm{R}_x\mathfrak{f}\|$ with R_x an orthogonal projection onto the $(n+1)$-dimensional linear subspace $\Sigma_x^{\perp}$.

By the union bound, for all $u,t>0$,

$$\mathbb{P}\left(\kappa_{\mathsf{aff}}(\mathfrak{f},x)\geq t\right)\leq \mathbb{P}\left(\|\mathfrak{f}\|\geq u\right)+\mathbb{P}\left(\|\mathrm{R}_x\mathfrak{f}\|\leq u/t\right). \tag{8.2}$$

We apply now Theorems 8.1 to the first term and 8.2 to the second. Thus for $u>5K\sqrt{N}$ and $t>0$,

$$\mathbb{P}(\kappa_{\mathsf{aff}}(\mathfrak{f},x)\geq t)\leq \exp(-u^2/(5K)^2)+\left(\frac{3u\rho}{t\sqrt{n+1}}\right)^{n+1}.$$

We set $u=5K\sqrt{N\ln(t)}$, so we get

$$\mathbb{P}\left(\kappa_{\mathsf{aff}}(\mathfrak{f},x)\geq t\right)\leq t^{-N}+\left(\frac{15K\rho\sqrt{N}}{\sqrt{n+1}}\right)^{n+1}\frac{\ln(t)^{\frac{n+1}{2}}}{t^{n+1}}$$

for $t\geq e$. The inequality $n+1\leq N$ finishes the proof. □

Theorem 8.3 immediately gives probabilistic bounds for the expressions $\mathbb{E}_{\mathfrak{x}\in[-a,a]^n}\left(\kappa_{\mathsf{aff}}(\mathfrak{f},\mathfrak{x})^n\right)$ and $\mathbb{E}_{\mathfrak{x}\in[-a,a]^n}\left(\kappa_{\mathsf{aff}}(\mathfrak{f},\mathfrak{x})^n\log^2\kappa_{\mathsf{aff}}(\mathfrak{f},\mathfrak{x})\right)$ for a random $\mathfrak{f}$. The two corollaries below, together with Theorems 6.2 and 7.2, give us the proof of the part (A) of Theorems 3.1 and 3.2.

Theorem 8.4 *Let $\mathfrak{f}\in\mathcal{P}_{n,d}$ be a dobro random polynomial with parameters K and ρ and $\alpha\in[1,n+1)$. Then*

$$\mathbb{E}_{\mathfrak{f}}\mathbb{E}_{\mathfrak{x}\in[-a,a]^n}\left(\kappa_{\mathsf{aff}}(\mathfrak{f},\mathfrak{x})^{\alpha}\right)\leq 4\frac{\alpha\sqrt{n+1}}{n+1-\alpha}\left(\frac{N}{n+1-\alpha}\right)^{\frac{n+1}{2}}(25K\rho)^{n+1}.$$

Corollary 8.1 *Let $\mathfrak{f}\in\mathcal{P}_{n,d}$ be a dobro random polynomial with parameters K and ρ. Then*

$$\mathbb{E}_{\mathfrak{f}}\mathbb{E}_{\mathfrak{x}\in[-a,a]^n}\left(\kappa_{\mathsf{aff}}(\mathfrak{f},\mathfrak{x})^n\right)\leq N^{\frac{n+1}{2}}2^{5n+\frac{3}{2}\log n+\frac{15}{2}}(K\rho)^{n+1}.$$

Corollary 8.2 *Let $\mathfrak{f}\in\mathcal{P}_{n,d}$ be a dobro random polynomial with parameters K and ρ. Then*

$$\mathbb{E}_{\mathfrak{f}}\mathbb{E}_{\mathfrak{x}\in[-a,a]^n}\left(\kappa_{\mathsf{aff}}(\mathfrak{f},\mathfrak{x})^n\log^2\kappa(\mathfrak{f},\mathfrak{x})\right)\leq N^{\frac{n+1}{2}}2^{6n+\frac{3}{2}\log n+12}(K\rho)^{n+1}.$$

Proof (Proof of Theorem 8.4) By the Fubini-Tonelli theorem,

$$\mathbb{E}_{\mathfrak{f}}\mathbb{E}_{\mathfrak{x}\in[-a,a]^n}\left(\kappa_{\mathsf{aff}}(\mathfrak{f},\mathfrak{x})^{\alpha}\right)=\mathbb{E}_{\mathfrak{x}\in[-a,a]^{\alpha}}\mathbb{E}_{\mathfrak{f}}\left(\kappa_{\mathsf{aff}}(\mathfrak{f},\mathfrak{x})^n\right)$$

so it is enough to have a uniform bound for

$$\mathbb{E}_{\mathfrak{f}}\left(\kappa_{\mathsf{aff}}(\mathfrak{f},x)^{\alpha}\right)=\int_1^{\infty}\mathbb{P}\left(\kappa_{\mathsf{aff}}(\mathfrak{f},x)^{\alpha}\geq t\right)\,\mathrm{d}t.$$

Now, by Theorem 8.3, this is bounded by

$$e^{\alpha}+2\left(\frac{N}{\alpha(n+1)}\right)^{\frac{n+1}{2}}(15K\rho)^{n+1}\int_1^{\infty}\frac{\ln(t)^{\frac{n+1}{2}}}{t^{\frac{n+1}{\alpha}}}\,\mathrm{d}t.$$

After the change of variables $t=e^{\frac{\alpha}{n+1-\alpha}s}$ the bound becomes

$$\begin{aligned}e^{\alpha}+2\frac{\alpha}{n+1-\alpha}&\left(\frac{N}{(n+1-\alpha)(n+1)}\right)^{\frac{n+1}{2}}(15K\rho)^{n+1}\int_1^{\infty}s^{\frac{n+1}{2}}e^{-s}\,\mathrm{d}s\\ &=e^{\alpha}+2\frac{\alpha}{n+1-\alpha}\left(\frac{N}{(n+1-\alpha)(n+1)}\right)^{\frac{n+1}{2}}\Gamma\left(\frac{n+3}{2}\right)(15K\rho)^{n+1},\end{aligned}$$

where Γ is Euler's Gamma function. We note that $e^{\alpha}\leq e^{n+1}$ and that, by the Stirling estimates,

$$\Gamma\left(\frac{n+3}{2}\right)\leq\sqrt{2\pi}\left(\frac{n+3}{2e}\right)^{\frac{n+2}{2}}\leq\sqrt{2\pi}\left(\frac{n+1}{e}\right)^{\frac{n+2}{2}}.$$

Combining all these inequalities, we obtain the desired upper bound. □

Proof (Proof of Corollary 8.1) We take $\alpha = n$ in Theorem 8.4. □

Proof (Proof of Corollary 8.2) Recall that $\log^2 y \leq 5\sqrt{y}$ for $y \geq 1$. Hence

$$\mathbb{E}_{\mathfrak{f}}\mathbb{E}_{\mathfrak{x}\in[-a,a]^n}\left(\kappa_{\mathsf{aff}}(\mathfrak{f},\mathfrak{x})^n\log^2\kappa(\mathfrak{f},\mathfrak{x})\right) \leq 2^{5/2}\mathbb{E}_{\mathfrak{f}}\mathbb{E}_{\mathfrak{x}\in[-a,a]^n}\left(\kappa_{\mathsf{aff}}(\mathfrak{f},\mathfrak{x})^{n+\frac{1}{2}}\right)$$

and the claim follows using Theorem 8.4 with $\alpha = n + \frac{1}{2}$. □

8.3 Smoothed Complexity Analysis

The tools used for our average complexity analysis yield also a smoothed complexity analysis (see [33] or [4, §2.2.7]). We provide this analysis following the lines of [20].

The main idea of smoothed complexity is to have a complexity measure interpolating between worst-case complexity and average-case complexity. More precisely, we are interested in the maximum —over $f \in \mathcal{P}_{n,d}$— of the average cost of the algorithm when the input polynomial has the form

$$\mathfrak{q}_\sigma := f + \sigma\|f\|\mathfrak{g} \tag{8.3}$$

with $\mathfrak{g} \in \mathcal{P}_{n,d}$ is a dobro random polynomials with parameters $K \geq 1$ and ρ, and $\sigma \in (0,\infty)$. Notice that the perturbation $\sigma\|f\|\mathfrak{g}$ of f is proportional to both σ and $\|f\|$.

The following lemma shows how Theorems 8.1 and 8.2 apply to this class of random polynomials.

Lemma 8.1 *Let $\mathfrak{q}_\sigma$ be as in* (8.3)*. Then for $t > 1 + \sigma\sqrt{N}$*

$$\mathbb{P}\left(\|\mathfrak{q}_\sigma\| \geq t\|f\|\right) \leq \exp\left(-\frac{(t-1)^2}{(\sigma 5K)^2}\right)$$

and for every $x \in \mathbb{R}^n$,

$$\mathbb{P}\left(\|\mathrm{R}_x\mathfrak{q}_\sigma\| \leq \varepsilon\right) \leq \left(3\rho\varepsilon/\left(\sigma\|f\|\sqrt{n+1}\right)\right)^{n+1}$$

where R_x is as in Corollary 5.1. □

Proof By the triangle inequality we have $\mathbb{P}(\|\mathfrak{q}_\sigma\| \geq t\|f\|) \leq \mathbb{P}(\|\mathfrak{g}\| \geq (t-1)/\sigma)$. Then we apply Theorem 8.1 which finishes the proof of the first claim. The second claim is a direct consequence of Theorem 8.2. □

As in the average case, this leads to a tail bound.

Theorem 8.5 *Let $\mathfrak{q}_\sigma$ be as in* (8.3) *and $x \in \mathbb{R}^n$. Then for $\sigma > 0$ and $t \geq e$,*

$$\mathbb{P}\left(\kappa_{\mathsf{aff}}(\mathfrak{q}_\sigma, x) \geq t\right) \leq 2\left(\frac{N}{n+1}\right)^{\frac{n+1}{2}}(15K\rho)^{n+1}\frac{\ln(t)^{\frac{n+1}{2}}}{t^{n+1}}\left(1+\frac{1}{\sigma}\right)^{n+1}.$$

Proof We proceed as in the proof of Theorem 8.3, but with Lemma 8.1 using $u = \|f\|(\sigma 5K\sqrt{N\ln(t)} + 1)$. This gives the desired bound arguing as in that proof after noticing that

$$u \leq \|f\|(1+\sigma)5K\sqrt{N\ln(t)}$$

which holds since $5K\sqrt{N\ln(t)} \geq 1$. □

As in the average case, Theorem 8.5 yields probabilistic bounds for both $\mathbb{E}_{\mathfrak{x}\in[-a,a]^n}\left(\kappa_{\mathsf{aff}}(\mathfrak{f},\mathfrak{x})^n\right)$ and $\mathbb{E}_{\mathfrak{x}\in[-a,a]^n}\left(\kappa_{\mathsf{aff}}(\mathfrak{f},\mathfrak{x})^n\log^2\kappa_{\mathsf{aff}}(f,\mathfrak{x})\right)$ for random $\mathfrak{f}$. The two corollaries below, together with Theorems 6.2 and 7.2, gives us the proof of the part (S) of Theorems 3.1 and 3.2.

Theorem 8.6 *Let $\mathfrak{q}_\sigma$ be as in* (8.3) *and $\alpha \in [1, n+1)$. Then for all $f \in \mathcal{P}_{n,d}$ and all $\sigma > 0$,*

$$\mathbb{E}_{\mathfrak{q}_\sigma}\mathbb{E}_{\mathfrak{x}\in[-a,a]^n}\left(\kappa_{\mathsf{aff}}(\mathfrak{q}_\sigma,\mathfrak{x})^\alpha\right) \leq 4\frac{\alpha\sqrt{n+1}}{n+1-\alpha}\left(\frac{N}{n+1-\alpha}\right)^{\frac{n+1}{2}}(25K\rho)^{n+1}\left(1+\frac{1}{\sigma}\right)^{n+1}.$$

Proof The proof is as that of Theorem 8.4, but using Theorem 8.5 instead of Theorem 8.3. □

Corollary 8.3 *Let $\mathfrak{q}_\sigma$ be as in* (8.3). *Then for all $f \in \mathcal{P}_{n,d}$ and all $\sigma > 0$,*

$$\mathbb{E}_{\mathfrak{q}_\sigma}\mathbb{E}_{\mathfrak{x}\in[-a,a]^n}\left(\kappa_{\mathsf{aff}}(\mathfrak{q}_\sigma,\mathfrak{x})^n\right) \le N^{\frac{n+1}{2}} 2^{5n+\frac{3}{2}\log n+\frac{15}{2}}(K\rho)^{n+1}\left(1+\frac{1}{\sigma}\right)^{n+1}.$$

Corollary 8.4 *Let $\mathfrak{q}_\sigma$ be as in* (8.3). *Then for all $f \in \mathcal{P}_{n,d}$ and all $\sigma > 0$,*

$$\mathbb{E}_{\mathfrak{q}_\sigma}\mathbb{E}_{\mathfrak{x}\in[-a,a]^n}\left(\kappa_{\mathsf{aff}}(\mathfrak{q}_\sigma,\mathfrak{x})^n \log^2\kappa(\mathfrak{f},\mathfrak{x})\right) \le N^{\frac{n+1}{2}} 2^{6n+\frac{3}{2}\log n+12}(K\rho)^{n+1}\left(1+\frac{1}{\sigma}\right)^{n+1}.$$

Proof (Proof of Corollaries 8.3 and 8.4) We do as in the proof of Corollaries 8.1 and 8.2 but using Theorem 8.6 instead of Theorem 8.4. □

Acknowledgements

We cordially thank Michael Burr and Elias Tsigaridas for useful discussions.

References

1. P. Bürgisser, F. Cucker, and M. Lotz. Smoothed analysis of complex conic condition numbers. *J. Math. Pures et Appl.*, 86:293–309, 2006.
2. P. Bürgisser, F. Cucker, and M. Lotz. The probability that a slightly perturbed numerical analysis problem is difficult. *Mathematics of Computation*, 77:1559–1583, 2008.
3. P. Bürgisser, F. Cucker, and J. Tonelli-Cueto. Computing the Homology of Semialgebraic Sets. II: General formulas. arXiv:1903.10710, March 2019.
4. Peter Bürgisser and Felipe Cucker. *Condition*, volume 349 of *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, Berlin, 2013.
5. Peter Bürgisser, Felipe Cucker, and Pierre Lairez. Computing the homology of basic semialgebraic sets in weak exponential time. *J. ACM*, 66(1):5:1–5:30, 2018.
6. Peter Bürgisser, Felipe Cucker, and Josué Tonelli-Cueto. Computing the Homology of Semialgebraic Sets. I: Lax Formulas. *Foundations of Computational Mathematics*, 20(1):71–118, 2020. On-line from May of 2019.
7. Michael Burr, Sung Woo Choi, Ben Galehouse, and Chee K. Yap. Complete subdivision algorithms II, Isotopic meshing of singular algebraic curves. *J. Symbolic Comput.*, 47(2):131–152, 2012.
8. Michael Burr, Felix Krahmer, and Chee Yap. Continuous amortization: A non-probabilistic adaptive analysis technique. *Electronic Colloquium on Computational Complexity*, Report. No. 136, 2009.
9. Michael A. Burr. Continuous amortization and extensions: with applications to bisection-based root isolation. *J. Symbolic Comput.*, 77:78–126, 2016.
10. Michael A. Burr, Shuhong Gao, and Elias P. Tsigaridas. The complexity of an adaptive subdivision method for approximating real curves. In *ISSAC'17—Proceedings of the 2017 ACM International Symposium on Symbolic and Algebraic Computation*, pages 61–68. ACM, New York, 2017.
11. Michael A. Burr, Shuhong Gao, and Elias P. Tsigaridas. The complexity of subdivision for diameter-distance tests. Available at `arXiv:1801.05864`, 2018.
12. F. Cucker and J. Peña. A primal-dual algorithm for solving polyhedral conic systems with a finite-precision machine. *SIAM J. Optim.*, 12:522–554, 2002.
13. Felipe Cucker. Approximate zeros and condition numbers. *J. Complexity*, 15(2):214–226, 1999.
14. Felipe Cucker, Alperen A. Ergür, and Josué Tonelli-Cueto. Plantinga-Vegter algorithm takes average polynomial time. In *Proceedings of the 2019 International Symposium on Symbolic and Algebraic Computation*, pages 114–121. ACM, New York, 2019.
15. Felipe Cucker, Teresa Krick, Gregorio Malajovich, and Mario Wschebor. A numerical algorithm for zero counting. I: Complexity and accuracy. *J. Complexity*, 24:582–605, 2008.
16. Felipe Cucker, Teresa Krick, Gregorio Malajovich, and Mario Wschebor. A numerical algorithm for zero counting. II: Distance to ill-posedness and smoothed analysis. *J. Fixed Point Theory Appl.*, 6:285–294, 2009.
17. Felipe Cucker, Teresa Krick, Gregorio Malajovich, and Mario Wschebor. A numerical algorithm for zero counting. III: Randomization and condition. *Adv. Applied Math.*, 48:215–248, 2012.
18. Felipe Cucker, Teresa Krick, and Michael Shub. Computing the Homology of Real Projective Sets. *Found. Comput. Math.*, 18:929–970, 2018.
19. James Demmel. The probability that a numerical analysis problem is difficult. *Math. Comp.*, 50:449–480, 1988.
20. Alperen A. Ergür, Grigoris Paouris, and J. Maurice Rojas. Probabilistic Condition Number Estimates for Real Polynomial Systems II: Structure and Smoothed Analysis. Available at `arXiv:1809.03626`, 2018.
21. Alperen A. Ergür, Grigoris Paouris, and J. Maurice Rojas. Probabilistic condition number estimates for real polynomial systems I: A broader family of distributions. *Found. Comput. Math.*, 19(1):131–157, 2019.
22. Stefan Funke. Of what use is floating-point arithmetic in computational geometry. In S. Albers, H. Alt, and S. Näher, editors, *Efficient Algorithms*, volume 5760 of *LNCS*, pages 341–354. Springer, 2009.
23. Benjamin T. Galehouse. *Topologically accurate meshing using domain subdivision techniques*. ProQuest LLC, Ann Arbor, MI, 2009. Thesis (Ph.D.)–New York University.

24. Herman H. Goldstine and John von Neumann. Numerical inverting matrices of high order, II. *Proc. Amer. Math. Soc.*, 2:188–202, 1951.
25. Nicholas Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, 1996.
26. Galyna Livshyts, Grigoris Paouris, and Peter Pivovarov. On sharp bounds for marginal densities of product measures. *Israel Journal of Mathematics*, 216(2):877–889, 2016.
27. Martin Lotz. On the volume of tubular neighborhoods of real algebraic varieties. *Proc. Amer. Math. Soc.*, 143(5):1875–1889, 2015.
28. Simon Plantinga and Gert Vegter. Isotopic approximation of implicit curves and surfaces. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '04, pages 245–254, New York, NY, USA, 2004. ACM.
29. Helmut Ratschek and Jon Rokne. *Computer methods for the range of functions*. Ellis Horwood Series: Mathematics and its Applications. Ellis Horwood Ltd., Chichester; Halsted Press [John Wiley & Sons, Inc.], New York, 1984.
30. Mark Rudelson and Roman Vershynin. The Littlewood-Offord problem and invertibility of random matrices. *Adv. Math.*, 218(2):600–633, 2008.
31. Mark Rudelson and Roman Vershynin. Small ball probabilities for linear images of high-dimensional distributions. *Int. Math. Res. Not. IMRN*, 19:9594–9617, 2015.
32. Stephen Smale. Complexity theory and numerical analysis. In A. Iserles, editor, *Acta Numerica*, pages 523–551. Cambridge University Press, 1997.
33. Daniel A. Spielman and Shang-Hua Teng. Smoothed analysis of algorithms. In *Proceedings of the International Congress of Mathematicians*, volume I, pages 597–606, 2002.
34. Josué Tonelli-Cueto. *Condition and Homology in Semialgebraic Geometry*. Doctoral thesis, Technische Universität Berlin, DepositOnce Repository, December 2019. http://dx.doi.org/10.14279/depositonce-9453.
35. Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47 of *Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge University Press, Cambridge, 2018.
36. Juan Xu and Chee Yap. Effective subdivision algorithm for isolating zeros of real systems of equations, with complexity analysis. In *ISSAC'19—Proceedings of the 2019 ACM International Symposium on Symbolic and Algebraic Computation*, pages 355–362. ACM, New York, 2019.
37. Chee Yap. Towards soft exact computation (invited talk). In *International Workshop on Computer Algebra in Scientific Computing*, pages 12–36. Springer, 2019.